Vysoká škola báňská – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra telekomunikační techniky

Přenos multiplexu s časovým dělením přes Ethernet

Time Division Multiplex Transport over Eehernet

Rok 2014

Adam Václavík

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra telekomunikační techniky

Zadání bakalářské práce

Student: **Adam Václavík**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2601R013 Telekomunikační technika

Téma: Přenos multiplexu s časovým dělením přes Ethernet
Time Division Multiplex Transport over Ethernet

Zásady pro vypracování:

Multiplex s časovým dělením (TDM) je v telekomunikacích nejčastěji zastoupen ve formě rozhraní E1 dle specifikace ITU-T G.703, které je využíváno např. u ISDN PRI. Již technologie ATM měla možnost emulovat E1 v paketově přepínané síti, dnes je ovšem požadavek na řešení v sítích IEEE 802.3. Cílem diplomanta je analyzovat současný stav realizace přenosu E1 přes Ethernet, navrhnout modelové řešení, naimplementovat v laboratorních podmínkách a provést měření kvalitativních parametrů vytvořeného E1 spoje. Jeden z možných způsobů je realizace pomocí Asterisk TDMoE technologie. V laboratoři IP telefonie bude k dispozici potřebné vybavení k realizaci, pomocí emulátoru sítě bude možné emulovat primární kvalitativní parametry jako zpoždění a ztrátovost a sledovat jejich vliv na chybovost zrealizované E1 trasy přes Ethernet. Výstupem by mělo být i nalezení limitních podmínek pro provozování diplomantem vytvořeného řešení.

1. TDM technologie a PDH, specifikace ITU-T G.703/704
2. Projekt Asterisk a konfigurace E1 rozhraní
3. Praktická realizace E1 tunelované přes Ethernet
4. Emulace ztrátovosti a zpoždění v IP a jejich vlivu na použitelnost tunelování E1
5. Vyhodnocení měření a zhodnocení dosažených výsledků

Seznam doporučené odborné literatury:

MADSEN L., MEGGELEN J., BRYANT R. *Asterisk: The Definitive Guide.* O'Reilly Media, 2011, 738p.
KOCUR Z., VODRAZKA J., MACEJKO P. *Low cost network emulator with ethernet and E1 interfaces.* Advances in electrical and electronic engineering, 2010, Vol. 8, No. 5, p. 134-136.
VOZŇÁK M., NEUMAN M., HOLÝ R. *Tunelování E1 přes síť CESNET2.* ČVUT Praha: Technická zpráva k řešení projektu, 31 str., 2006.
VOZNAK M., BENES J. *TDM over IP Solution.* In Proc. Research in Telecommunication Technologies (RTT2006), ,Brno University of Technology, 2006, p.382-384.

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **doc. Ing. Miroslav Vozňák, Ph.D.**

Datum zadání: 16.11.2012

Datum odevzdání: 07.05.2013

prof. RNDr. Vladimír Vašinek, CSc.
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlášení studenta

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě dne: *6. května 2014*

……
podpis studenta

Poděkování

Rád bych poděkoval doc. Ing. Miroslavu Vozňákovi, Ph.D. za odbornou pomoc a konzultaci při vytváření této bakalářské práce. Také bych chtěl poděkovat své rodině za podporu a zázemí při tvorbě.

Abstrakt

Multiplex s časovým dělením (TDM) je v telekomunikacích nejčastěji zastoupen ve formě rozhraní E1 dle specifikace ITU-T G.703, které je využíváno např. u ISDN PRI. V této bakalářské práci využiji open-source technologie Asterisk TDMoE, k přenosu časového multiplexu přes rozhraní Ethernet a telefonní karty od společnosti Digium. V laboratořích s potřebným vybavením k realizaci, budu pomocí emulátoru sítě emulovat primární kvalitativní parametry jako je Latence, Bit Error Rating a Jitter a sledovat jejich vliv na chybovost zrealizované E1 trasy přes Ethernet. V závěru budou nalezeny také limitní podmínky pro provoz takto vytvořené trasy.

Klíčová slova

Asterisk; DigiumTE110P; TDM; E1; TDMoE;

Abstract

Time Division Multiplex (TDM) is a telecommunications frequently represented as E1 interface specified by ITU-T G.703, which is used eg for ISDN PRI. In this thesis I will use open-source technology Asterisk TDMoE to time multiplex transmission via Ethernet. For this purpose use a phone card from Digium company. In laboratories with the necessary equipment to implement, I'm using an emulator to emulate a network of primary qualitative parameters such as latency, Bit Error Rating and Jitter and monitor their impact on the error E1 routes implemented over Ethernet. In the end they also found the limit conditions for the operation of the route follows.

Key words

Asterisk; DigiumTE110P; TDM; DAHDI; E1; TDMoE;

Seznam použitých zkratek

Zkratka	Význam
***CLI>**	Command Line Interface
AMI	Alternate Mark Inversion code
API	Application Programming Interface
BER	Bit Error Rating
CRC4	Cyklický redundantní součet
PDH	Plesiochronní Digitální Hierarchie
RAI	Remote Alarm Indication
REC	Reload alarm telefonní karty
RED	Červený alarm telefonní karty
TDM	Time Division Multiplex
TDMoE	Time Division Multiplexing over Ethernet
VoIP	Voice over Internet Protocol
YEL	Žlutý alarm telefonní karty

Obsah

Úvod

Dávno pryč jsou doby, kdy nám stačili ke komunikaci mezi dvěma účastníky klasické telefony nebo faxy. S nástupem digitální telefonie a prostupováním internetu prakticky všude, se objevily nápady a poté i realizace propojení internetu a volání. Tento nápad dostal jméno VoIP (Voice over IP) a představuje paketizovaný přenos hlasu skrze IP protokol. Ovšem nic není jednoduché a každé řešení s sebou přináší i nové problémy. U VoIP takovým problémem může být přenos dat skrze navzájem nekompatibilní sítě. Abychom tento problém mohli odstranit a přenést hovor i přes takovéto sítě, využijeme síťového tunelování. Tunelováním sítí je myšleno vytvoření datového kanálu mezi dvěma počítači v internetové síti. Takto vznikne ze dvou vzdálených rozdílných sítí námi vytvořená síť uvnitř veřejné internetové sítě, která je zdánlivě homogenní. Řešení pomocí tunelování sítě nám přináší mnohem více výhod. Asi nejpodstatnější výhodou je nízká nákladovost pro firmu s několika divizemi, kdy každá divize má již existující vlastní síť. Pro firemní společnost se sice naskytuje možnost objednat řešení v podobě VPN u svého poskytovatele telekomunikačních služeb, ale tento způsob je zbytečně finančně nákladný. Další možnou, téměř bezplatnou metodou, kterou nabízí téma mé práce, je propojení ústředen pomocí již existující paketové páteřní sítě, kdy využijeme stávající firemní hardware, pouze upravíme jeho nastavení. Nutnou podmínkou realizace tohoto řešení je existence vlastní pobočkové ústředny PBX v každé divizi firmy.

Dalším způsobem, jak využít tunelování sítí pro přenos rámců E1, je poskytovat zabezpečenou komunikaci přes nezabezpečenou síť. Pro tento účel využijeme datového tunelu mezi dvěma pobočkami k přenosu šifrovaného hovoru skrze veřejnou nešifrovanou síť.

Cílem mé práce je navrhnout modelové řešení pro tunelování TDM přes Ethernet popsané v druhé kapitole. V následující kapitole se zabývám jeho praktickou realizací v laboratorních podmínkách a úvodem k měření, kde vysvětluji, jak bude měření probíhat. Ve čtvrté kapitole se zabývám měřením kvalitativních parametrů vytvořeného E1 spoje. Pomocí emulátoru sítě zde emuluji primární kvalitativní parametry, jako zpoždění BER a Jitter a popisuji jejich vliv na chybovost zrealizované E1 trasy přes Ethernet. Výstupem je i nalezení limitních podmínek pro provozování vytvořeného řešení.

1 Vysvětlení pojmů a užitých technologií

Tato kapitola slouží k přiblížení a vysvětlení jednotlivých technologií a odborných termínů použitých v mé práci.

1.1 Síťové emulace

Při výstavbě nebo návrhu síťového spojení předchází samotné realizaci, fáze projektování a s tím i spojená simulace provozu této sítě. K tomuto účelu využíváme síťových emulací. Jedná se tedy o techniku, kdy jsou vlastnosti existujícího nebo plánovaného síťového spojení simulovány za účelem posouzení výkonnosti, předvídání chyb, předvídání dopadu změn ve vedení nebo jinak optimalizovat a zkoumat síťové spojení. To se provádí v laboratorním prostředí na zkušební síti s parametry sítě plánované. Lze tedy operativně měnit tok paketů tak, aby to odpovídalo reálnému zatížení sítě. Abychom plně otestovali síť, je žádoucí aby bylo možno pomocí síťových emulací simulovat nejčastější chyby při přenosu jako je ztrátovost rámců, duplikace rámců, zpoždění, jitter, simulace šířky pásma, simulace zahlcení, poškozování obsahu.

Z praxe víme, že sítě nejsou dokonalé, ať už soukromé nebo veřejné. Běžně se setkáváme s výpadky, zahazováním paketů apod. Cílem síťové emulace je vytvořit prostředí, ve kterém mohou uživatelé hodnotit a zkoumat své produkty, hodnotit jejich výkon nebo stabilitu a to v bezpečném a sebou kontrolovaném prostředí.

Emulace se liší od simulace v tom, že síťový emulátor se zdá být sítí. Lze k němu připojit koncová zařízená, která se pak chovají, jako vy byla připojena k reálné síti. Sítový emulátor emuluje síť, která spojuje koncové systémy.

Stejně jako u všech produktů, síťové emulátory se vyskytují v různých formách a poskytují různé možnosti. Může se jednat o programy běžící na jednom počítači nebo to mohu být sofistikovaná hardwarová zařízení připojující se paralelně k síti.

Hardwarové produkty síťové emulace jsou vyráběny za použití speciálních komponentů sestavených jednoznačně a za jediným účelem a to zpracování různých síťových situací. Nejedná se pouze o univerzální počítačový hardware doplněn o síťový emulační software. Celá platforma je účelově vyvíjena k síťové emulaci. Takto vyvinuté emulátory nabízejí největší výkon a přesnost při testování.

Softwarové produkty jsou programy pracující na různých zařízeních například počítačích a pod různými operačními systémy. Jejich hlavní nevýhodou oproti hardwarovým emulátorům je nižší přesnost a ne zcela optimalizovaný výpočetní výkon, nebo absence pokročilých funkcí. Na druhou stranu jsou mnohem levnější než jejich hardwarová konkurence. Na operačních systémech Windows můžeme použít síťový emulátor ZTI NetDisturb jenž lze nainstalovat na PC s minimálně dvěma síťovými kartami. Program umožňuje v rámci laboratoře vytvořit totožné podmínky a přenosové parametry, jaké se vyskytují v reálné síti.

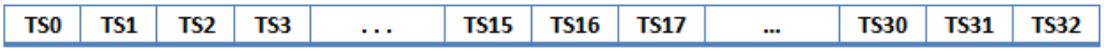
Emulovat můžeme ztrátovost rámců, duplikace rámců, zpoždění, jitter, simulace šířky pásma, simulace zahlcení, poškozování obsahu.

V současných linuxových systémech lze využít nástroje Netem. Tento software poskytuje funkce síťového emulátoru pro testování protokolů tím, že napodobuje vlastnosti WAN sítí. Aktuální verze 2.6 emuluje proměnné zpoždění, ztrátu, zdvojování a re-uspořádání. Software Netem je podporován na systémech Fedora , OpenSuse , Gentoo , Debian , Mandriva , Ubuntu přímo v jádře díky iproute2. Netem neobsahuje grafické rozhraní, proto se řídí pouze pomocí příkazového řádku. Více o softwaru NetEM a síťových emulacích zde [1]

1.2 Rozhraní E1

Rozhraní E1 představuje první řád v evropské plesiochronní digitální hierarchie (PDH). Jedná se o zdigitalizovaný analogový signál, který je multiplexován do jednoho rámce. Tento rámec se skládá z 32 Timeslotů (TS0 – TS31). Jmenovitá délka jednoho rámce je 125μs a opakovací kmitočet je 8kHz. Každý tento Timeslot (Timeslot ≡ Kanálový interval) se skládá z 8 bitů a má délku 3,9 μs viz obrázek 1.2. Odsud si lze spočítat přenosovou rychlost E1:

$$E1 = 8\ bitů \cdot 32\ TS \cdot 8000\ Hz = 2048\ kbit/s \qquad (1.1)$$



Obrázek 1.1: *Strukruta E1 rámce*

Podobně lze dopočítat přenosovou rychlost jednoho Timeslotu:

$$TS = 8\ bitů \cdot 8000\ Hz = 64\ kbit/s \qquad (1.2)$$

Obrázek 1.2: *Struktura jednoho timeslotu*

Aby bylo možné KI směrovat po síti do cíle určení, bylo třeba přenášet kromě dat i signalizaci. Pro tu je vyhrazen 16. kanálový interval (obrázek 1.3) v každém rámci. Přenáší se však v každém druhém rámci, tedy v R1, R3, R5, …. Multirámcová signalizace umístěná v R0 má složení 0000 a označuje se zkratkou MFAS (Multirame Aligment Signal). Bity kde tato signalizace není, jsou označeny v rámci R0 jako NONMFAS.

Rámec	S1	S2	S3	S4	S5	S6	S7	S8
R0	0	0	0	0	X	Y	X	X
R1 – R15	0	0	0	0	a	b	c	d

Obrázek 1.3: *16. kanálový interval*

X – digitální kanály s přenosovou rychlostí 500 bit/s
Y – poplachový signál ztráty MFAS, přenosová rychlost je 500 bit/s
a,b – signalizační kanály
c – nastaven na stálou hodnotu log 0
d – nastaven na stálou hodnotu log 1

Stejně tak bylo třeba zajistit správné pořadí rámců při každém demultiplexování a místo pro přenos služebních informací, které zajišťují správný chod sítě. To zase zajišťoval 0 kanálový interval. Obsah nultého Timeslotu se lišil podle umístění rámce – KI v sudém rámci obsahoval synchroskupinu a KI v lichém rámci synchroskupinu neobsahoval. Signál rámcové synchronizace má složení 0011011 a je umístěn v sudých rámcích na pozicích bitů S2 - S8 nultého kanálového intervalu R0. Synchroskupina 0011011 se označuje zkratkou FAS(Frame Alignment Signal), V lichých rámcích je na místě S2 symbol log 1. Zmenšuje se tím pravděpodobnost náhodného napodobení rámcové synchronizace, protože se na místě S2 pravidelně střídá nula a jednička. Ke ztrátě rámcového souběhu dojde jestliže příjmaná synchroskupina je přijata 3x za sebou s poruchou alespoň jednoho bitu. K obnově rámcového souběhu dojde, jestliže byla přijata bezchybná synchroskupina v rámci n, v následujícím rámci n+1 synchroskupina chybí, v rámci n+2 je znovu přítomna bezchybná synchroskupina. Symbol X na místě S1 v každém rámci je využíván pro CRC (Cyklický Redundantní Kód) hašovací funkci užívanou k detekci chyb během přenosu. Symbol označený Y (S3 v lichých rámcích TS0) může být využit pro přenos havarijní signalizace ke vzdálené stanici. V bezporuchovém stavu má hodnotu log 0, při hlášení poruchy hodnotu log 1. Symboly Z, U, V1, V2, V3 představují digitální kanál 4 kbit/s, použitelný podle CCITT v národních sítích. Více o rámci E1 zde [1] [3]

Rámec	S1	S2	S3	S4	S5	S6	S7	S8
R0, sudé rámce	X	0	0	1	1	0	1	1
R1, liché rámce	X	1	Y	Z	U	V1	V2	V3

Obrázek 1.4: *Nultý kanálový interval*

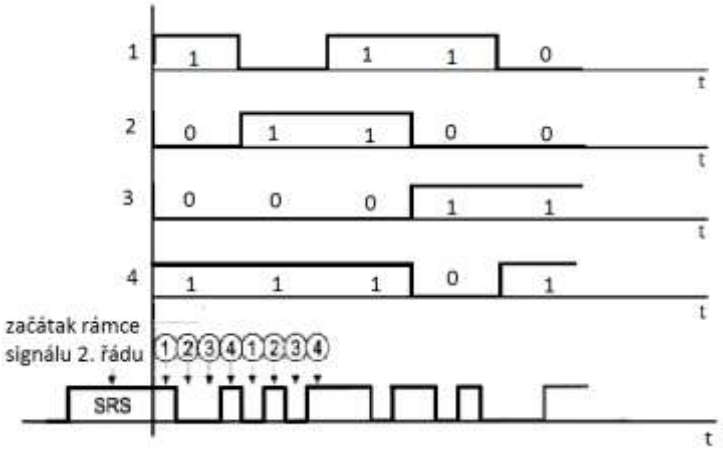
1.3 PDH

PDH je jedna z prvních digitálních hierarchií používaných pro přenos telefonního signálu. Novějšími typy jsou synchronní SDH hierarchie, která je v dnešní době velmi rozšířená a prakticky nahradila PDH nebo, ne tolik úspěšná, avšak velmi drahá technologie ATM. Důvodem pro vznik digitálních hierarchií a jejich standardizaci byla potřeba propojit digitální telefonní ústředny a dále pak zvyšovat rychlost digitálního přenosu po lince mezi těmito ústřednami. Cílem bylo multiplexovat digitální toky jako je audio, video, data a podobně ve vysokorychlostní signál, který by bylo možné přenášet jediným spojem. Tato multiplexace se prováděla buď prokládáním bit po bitu, nebo po kódových skupinách (rámcích velkých každý 8 bitu) nebo po celých rámcích, buňkách, paketech. Prokládání bylo řízeno buď volně, kdy

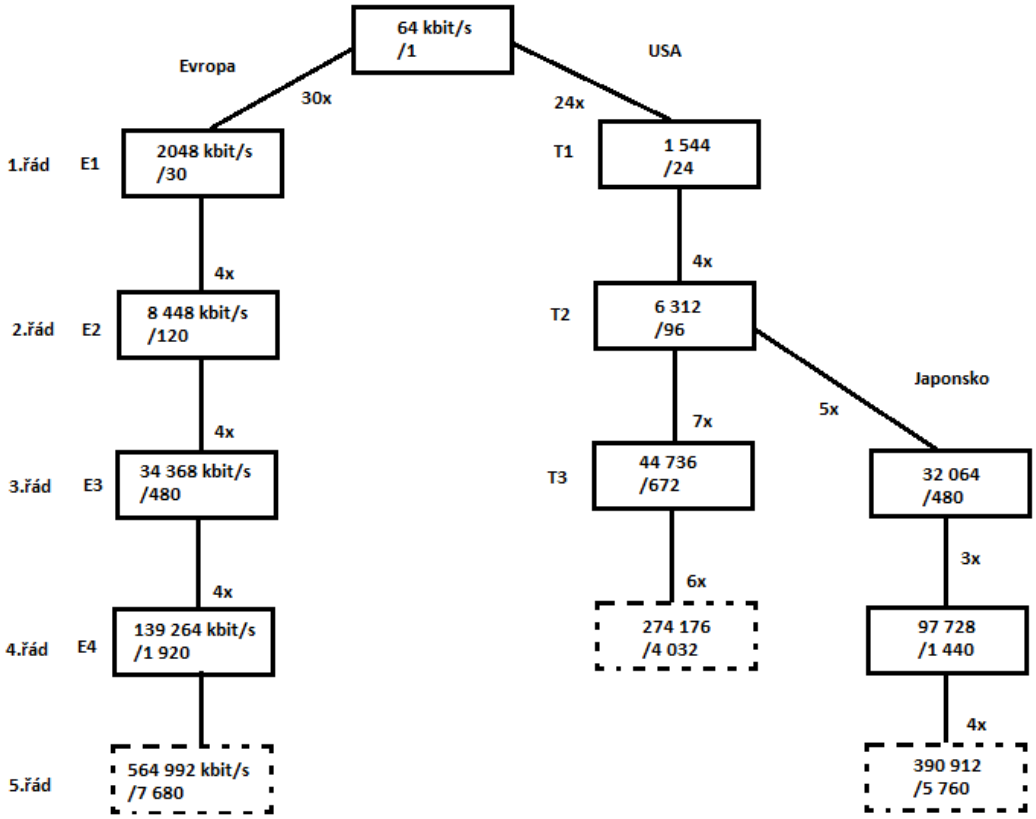
docházelo k minimálnímu časovému zpoždění anebo bylo řízené, při kterém se zachovávala informace o umístění rámce příspěvkového signálu. Dalším cílem bylo přenášet větší počet informačních kanálů, než umožňovala PCM 1. řádu.

PDH zkratka znamená plesiochronní digitální hierarchie. Slovo plesiochronní je odvozeno z řeckého plēsios a v překladu znamená téměř a chronos znamenající čas; odkazuje na fakt, že PDH linky běží téměř synchronizovaně. Jednotlivé zařízení, jsou tedy řízeny vlastními časovými základnami, což vede ke kolísání okamžitých hodnot taktovacího kmitočtu okolo jmenovité hodnoty.

Jedná se o PCM multiplexní zařízení n-tého řádu, které sdružuje čtyři digitální signály n-1 řádu. A to tak, že na vysílací straně bere postupně jeden bit z prvního proudu signálu, jeden bit z druhého proudu signálu, jeden bit z třetího proudu signálu a jeden bit ze čtvrtého proudu signálu viz obrázek 1.6: Sdružování signálu. Navíc přidává dodatečné bity pro synchronizaci a výplňkové bity. Pro konkrétní představu uvedu příklad, kdy pro PDH druhého řádu použijeme čtyři E1 rámce. Ve výsledku tedy dostaneme přenosovou rychlost takovéhoto multiplexu 8448 kbit/s a 120 použitelných kanálů pro data, viz obrázek 1.5. Pro Evropu se jedná o multiplex vždy čtyř nižších řádů, ale v USA a Japonsku jsou mezi jednotlivými řády používány různé násobky.



Obrázek 1.5: *Multiplexování signálu*



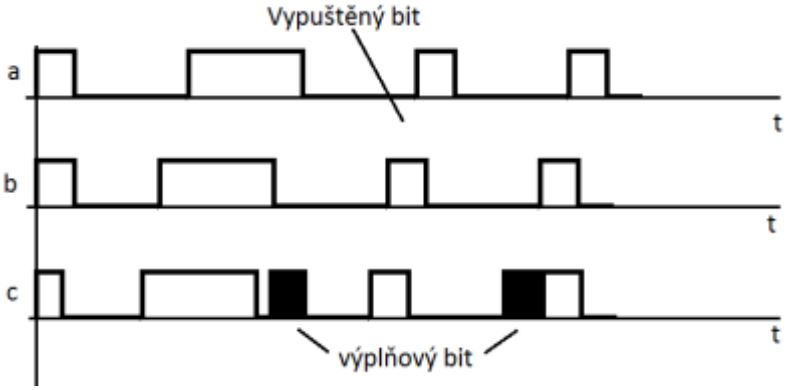
Obrázek 1.6: *Sdružování signálů*

Při multiplexování prokládáme sdružované signály bit po bitu do rámce vyššího řádu, aniž by byl definován jakýkoliv vztah mezi singály nižšího řádu a rámcem signálu vyššího řádu. Toto prokládání je prováděno volně, tedy každý multiplexovaný signál může mít jinou přenosovou rychlost. Tyto nepřesnosti jsou poté vyrovnávány takzvaným stuffingem. Schéma stuffingu je na obrázku 1.7. Ten je kladný, záporný nebo kombinovaný.

Při kladném stuffingu používaném v Evropě je taktovací kmitočet časové základny aparátu vyššího řádu vyšší, než je maximální okamžitá rychlost přenosu příspěvkového signálu (rezerva kvůli možné toleranci kmitočtů). Stuffingové bity jsou nevyužity a tvoří pouze výplň

U záporného stuffingu je pro signál v rámci rezervován právě takový počet míst, odpovídající nejnižší možné přenosové rychlosti signálu. U záporného stuffingu jsou tedy stuffingové bity obsazeny daty.

Kombinovaný, nebo jinak řečeno oboustranný stuffing znamená, že v multiplexovaném signálu vyššího řádu je vyhrazen právě takový počet bitů, odpovídající velikosti přispívajícího signálu. Odchylky v rychlosti jsou jak kladné tak i záporné a podle potřeby se používá kladný nebo záporný stuffing.



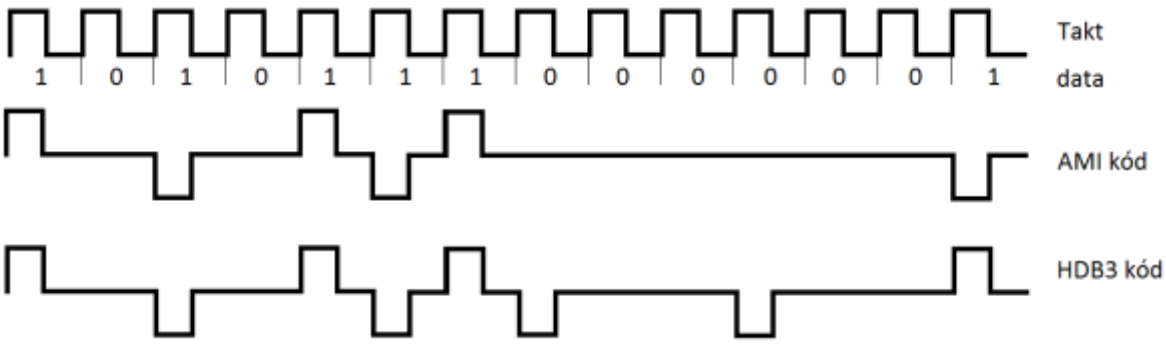
Obrázek 1.7: *Stuffing*

Jak je vidět, PDH není dokonalý systém. Mezi jeho největší nevýhody, kvůli kterým byl nahrazen systémem SDH patří omezená přenosová rychlost a problematické vydělování digitálních skupin signálů v uzlových bodech sítě, kdy je nutné provézt několikanásobné demultiplexování.[4]

1.4 HDB3 kódování

Tento kód vychází ze staršího AMI (Alternate Mark Inversion) kódu. Ten kóduje binární "0" jako nulovou úroveň napětí, stejně jako v unipolárním kódování a binární "1" kóduje střídavě kladným a záporným napětím.

Kód HDB3, nebo také obecněji označován jako (HDBn), kde n určuje nejvyšší počet nul po sobě jdoucích. Impulsy původního vysílaného binárního signálu se v tomto kódu kódují střídajícími se impulsy +B a -B, nuly se pak převádějí opět jako nuly. Ovšem v případě, že v původním vysílaném binárním signálu následují za sebou 4 a více nul, zavede se do výsledného signálu HDB3 takzvané porušení bipolarity (tj. následují za sebou dva impulsy stejné polarity a význam symbolů se mění). Takovýto bit porušující bipolaritu je označen písmenem V (Violation). Pokud má předchozí impuls inverzní polaritu oproti poslednímu porušení bipolarity, převádí se první nula ze skupiny čtyř nul jako nula. Pokud má polaritu stejnou, vyjádří se první nula ze skupiny impulsem +B nebo -B, který neporušuje bipolaritu. Takovýmto způsobem se zajistí, aby po sobě následující porušení bipolarity měly pokaždé opačnou polaritu, tudíž nevzniká SS složka (stejnosměrná složka) přenášeného signálu. To je velmi důležité, protože odstranění SS složky, která by posouvala úrovně signálu za translátory je důvod, proč musí být přenášen bipolární a ne unipolární signál. Další nuly (tj. druhá a třetí nula) ze skupiny čtyř nul kódu HDB3 se vždy převádějí stejně, tedy jako nuly. Nula na čtvrtém místě se vždy vyjádří impulsem V (Violation), jehož polarita je stejná s posledním impulsem předchozím (porušení bipolarity).



Obrázek 1.8: *HDB3 kódování*

Z obrázku 1.8 jde vidět, že může být maximálně posloupnost 3 nul za sebou a posloupnost 4 a více nul je nahrazena skupinou 000V nebo B00V, kde V je voleno tak aby se narušilo pravidelné střídání kladných a záporných impulsů.

HDB3 byl zaveden kvůli tomu, aby se eliminovaly dlouhé toky 0 v přenosu. Mezi jeho další výhody patří odstraňování SS složky a nezmenšování šířky pásma. Takovéto kódy nahradily AMI v moderních distribučních sítích.[5]

1.5 Signalizace DSS1

Signalizace je v telekomunikačních systémech důležitá, protože právě díky ní jsou pakety směrovány po síti až do cíle určení. Signalizace je tedy nezbytnou podmínkou pro funkčnost jakékoliv telekomunikační sítě. Úkolem signalizace je navazovat spojení mezi dvěma zařízeními, ať už účastník-ústředna nebo ústředna-ústředna, dále držet dohled nad vytvořeným spojením a řídit rozpad spojení. V závislosti na typu zařízení, mezi kterými má signalizace vést a které má spojovat, se užívá různých typů signalizací (SS7 – síťová signalizace, DSS1 – účastnická signalizace/signalizace pobočkových ústředen, CAS – digitalizovaná analogová signalizace, Q-sig – v privátních sítích, apod.). V roce 1993 vydala mezivládní organizace ITU-T oficiální definici Signalizace znějící takto:

Signalizace (signalling) - výměna informací týkajících se sestavování a řízení spojení v telekomunikačních sítích a také výměna informací při dohlížení nad těmito sítěmi.

Vzhledem k užití ISDN telefonů nás bude nejvíce zajímat princip a průběh signalizace DSS1. Jedná se o v dnešní době jediný používaný signalizační systém mezi koncovým účastníkem ISDN a telefonní ústřednou. Zásluhu na jeho vzniku si připsala organizace ITU-T.

Signalizační systém DSS1 je přístupový signalizační systém pro spojení typu bod - bod. Není proto nutné na úrovni třetí vrstvy přenášet údaje potřebné pro směrování v síti. Nad spojovou vrstvou je přenášena již přímo signalizační zpráva. Protokol využívá spodních tří vrstev referenčního modelu OSI. Více v doporučení Q. 933 viz zde[6]

1.5.1 **Fyzická vrstva:**

Fyzická vrstva je prezentována tokem bitů se stanovenou rychlostí. Na úrovni první, jak můžeme vidět na obrázku 1.9, fyzické vrstvy, se k přenosu signalizace využívá speciálních signalizačních D kanálů. Přenosové rychlosti závisí na užité ISDN přípojce:

BRI (Basic-Rate-Access) – D kanál má rychlost 16kb/s. Tento kanál je využíván jak k přenosu signalizačních zpráv, tak k přenosu uživatelských dat. Nelze však D kanálem přenášek obojí najednou. To znamená, že ve chvíli kdy kanálem neproudí signalizační zprávy, začne přenášet uživatelská data. Ovšem ve chvíli, kdy je třeba přenést signalizační zprávu, je tok uživatelských dat přerušen a posílá se signalizace, jenž má vyšší prioritu než data.

PRI (Primary-Rate-Access) – Zde je přenosová rychlost D kanálu 64 kb/s. přípojka je realizována multiplexem PCM 1. Řádu, tudíž se D kanál přenáší v 16. kanálovém intervalu.

1.5.2 **Spojová vrstva:**

Na úrovni spojové vrstvy L2, znázorněné na obrázku 1.9, je pro přenos signalizace doporučením předepsáno použití linkového protokolu LAPD (Link-Access-Procedure-on-the-D-Channel). Jedná se o bitově orientovaný linkový protokol, který vznikl dílčími úpravami linkového protokolu HDLC pocházejícího z paketových sítí. Jeho úkolem je zabezpečení přenášené signalizace a dat v D kanále proti přenosovým chybám.

1.5.3 **Síťová vrstva:**

Na úrovni síťové vrstvy jsou poskytovány služby potřebné pro účastnickou přípojku (např. BRI) a zasílány zprávy o řízení nebo ovládání služeb. Každá zpráva síťové vrstvy se skládá ze záhlaví a doplňujících prvků. Záhlaví zprávy je složeno ze tří částí:

Protocol Discriminator (Typ protokolu) vyjadřuje, o jaký typ signalizační zprávy se jedná a má hodnotu 0x08

Call Reference označuje všechny zprávy patřící jedné signalizační aktivitě (např. volání). CR se přidělí hned na začátku spojení a je použit ve všech signalizačních zprávách mezi terminálem a ústřednou až do ukončení spojení.

Message Type vyjadřuje určitou proceduru při řízení spojení. Tyto zprávy dělíme do několika skupin podle významu zprávy:

Sestavování spojení: ALERTING, CALL PROCEEDING, CONNECT, CONNECT ACKNOWLEDGE, PROGRESS, SETUP, SETUP ACKNOWLEDGE

Rušení spojení: DISCONNECT, RELEASE, RELEASE CMPLETE, RESTART, RESTART ACKNOWLEDGE

Doplňkové služby/dohledové: RESUME, RESUME ACKNOWLEDGE, RESUME REJECT, SUSPEND, SUSPEND ACKNOWLEDGE, SUSPEND REJECT, USER INFORMATION

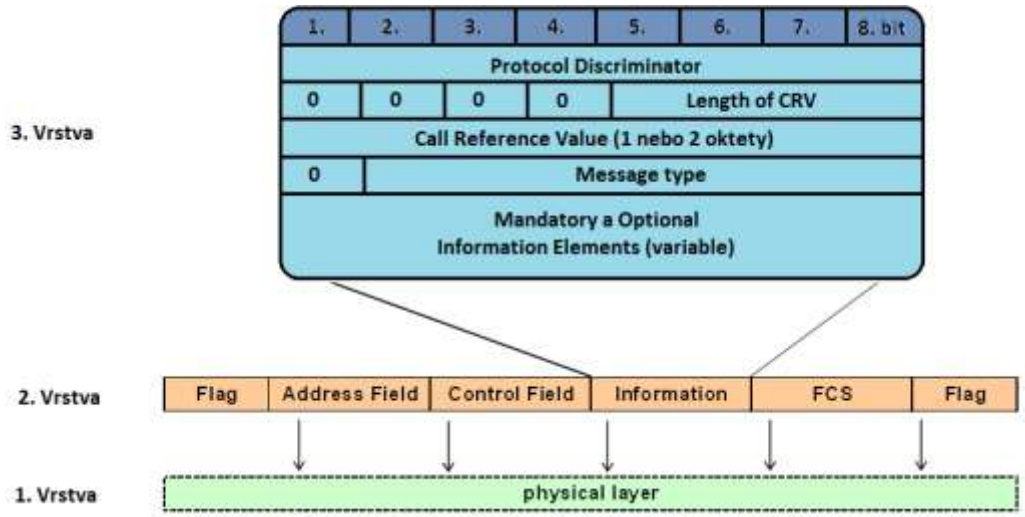
Různé zprávy: SEGMENT, CONGESTION CONTROL, INFORMATION, FACILITY, NOTIFY, STATUS, STATUS ENQUIRY

dále zprávy pro "globální"řízení: RESTART, RESTART ACKNOWLEDGE, STATUS

zprávy pro doplňkové služby nezávislé na spojení

zprávy pro portabilitu terminálů

zprávy pro přenos informace mezi uživateli



Obrázek 1.9: *Popis užitých vrstev v protokolu DSS1*

Každý typ zprávy má povinné (Mandatory) a volitelné (Optional) informační prvky:

• Bearer Capability identifikuje požadavky přenosu pro B-kanál

• Cause nese důvody rozpojení nebo nedokončeného volání,

• Channel Identification identifikuje typ a počet B-kanálů,

• Progress Indicator oznamuje stav volání, nese např. informaci, že se v informačním kanále přenáší nějaká informace anebo zařízení není ISDN (inband information available nebo non-ISDN),

• Calling Party Number nese číslo volajícího

• Called Party Number přenáší číslo volajícího

1.6 TDM

(Time division multiplex) neboli česky multiplex s časovým dělením byl vyvinut francouzským inženýrem Émilé Baudotem nejprve pro použití v telegrafii na směrování více přenosů současně po jednom přenosovém kanálu. Plné využití však našel až o několik desítek let později v digitálních telefonních sítích ve druhé polovině 20. století.

Pro realizaci časového multiplexu je nutnou podmínkou, aby signál, který přenáší informaci a který my budeme dále multiplexovat, měl maximální frekvenční rozsah 300 – 3400Hz. Tento signál se dále upraví na digitální pomocí A/D převodníku se vzorkovací frekvencí 8kHz (kvůli dodržení *Nyquistova* kriteria). Signál je podroben PCM modulaci s 256 kvantovacími úrovněmi, 128 pro kladnou půlvlnu a 128 pro zápornou půlvlnu. Vytváří se tedy 8 bitové slovo.

Princip přenosu v časovém multiplexu spočívá v přenosu více digitálních signálů jedním společným přenosovým médiem. Toto seskupení signálů se jmenuje Timeslot. Každý jednotlivý signál je vysílán pouze v krátkých pevně definovaných časových intervalech, trvajících každý 3,9 ms. Pokud budeme mít čtyři signály A,B,C,D, bude vysílání pomocí časového multiplexu vypadat následovně: ABCDABCDABCDABCD… a tak stále dokola. Důležité bylo, aby při demultiplexování zpět na jednotlivé signály bylo poznat, kde jednotlivý timeslot začíná a kde končí. To se vyřešilo přidáním speciálního signálu s přesně definovaným obsahem na začátek každého Timeslotu, tak aby označoval jeho začátek. Protože všechny kanály trvají stejně dlouhou dobu, stačí na přijímající straně najít tento speciální synchronizační signál a odpočítat potřebnou dobu do začátku hledaného signálu. Každý Timeslot tedy vypadá takto: $ABCD$ABCD$ABCD$ABCD… kde $ označuje synchronizační signál. To vše se reálně provádí pro 30 signálů v každém Timeslotu.

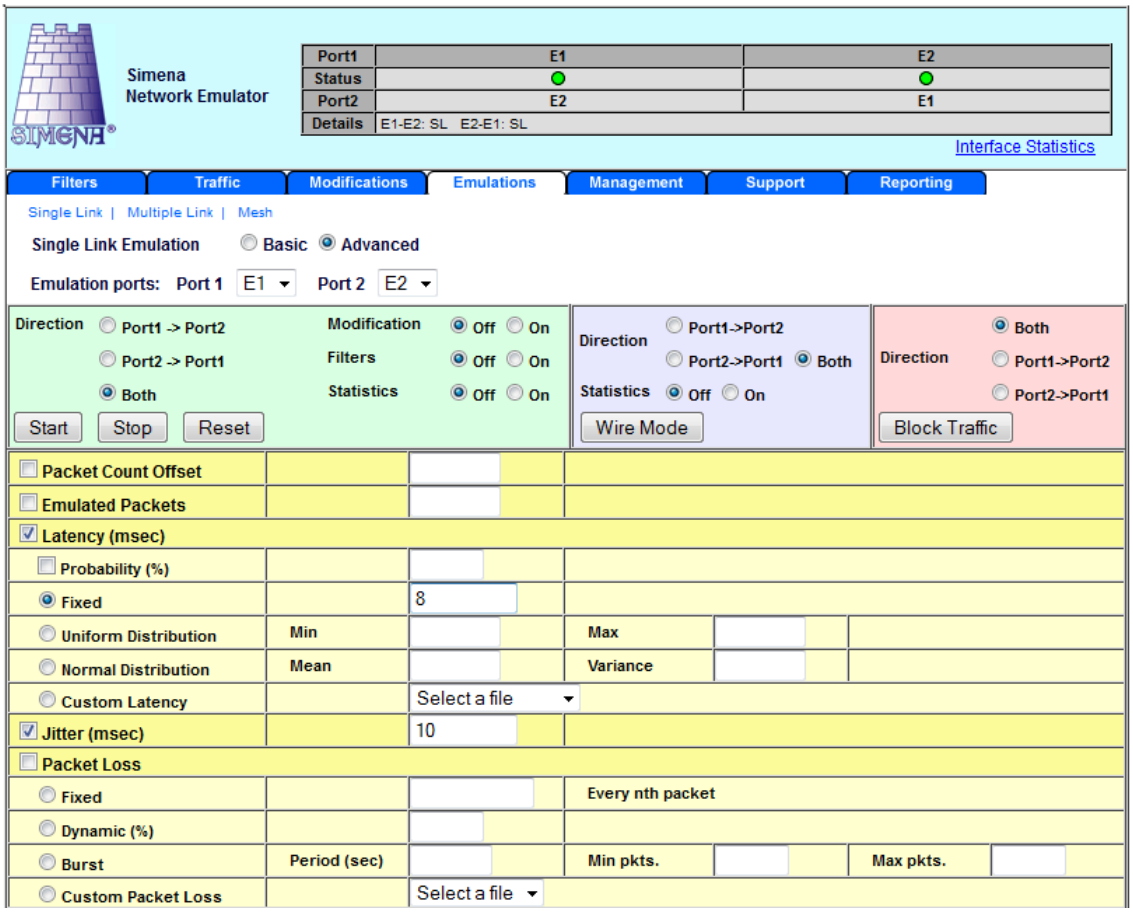
Časového multiplexu se využívá převážně v telefonii, jak již plyne z první podmínky. Například v ISDN, Plesiochronní Digitální Hierarchii (PDH) nebo Synchronní Digitální Hierarchii (SDH/SONET)[7]

1.7 Simena NE1000

Síťový emulátor Simena umožňuje vývojářům softwaru a síťovým inženýrům, testovat ve svém testovacím prostředí, aby se zjistilo, jak je jejich produkt nebo služba kvalitní na základě několika síťových podmínek, jako je rychlost, latence, BER, atd. Emulátor tyto podmínky napodobí tím, že zachytí a zpracuje datové pakety transparentně; připojená zařízení fungují jako by byla připojena k produkční síti. Síťové Emulátory můžou být použity s jakýmkoli síťovým protokolem (IP, IPX, AppleTalk, atd.) a síťovým rozhraním. Vzhledem k tomu, že pracují na linkové vrstvě, nevyžadují žádné změny konfigurace sítě v klientských pracovních stanicích nebo aplikačních serverech.

Simena NE1000 je model střední úrovně se systémem na bázi Linuxu. Je osazen dvěma Gigabit Ethernet porty pro emulaci a jedním Fast Ethernet portem pro správu. Díky jeho rozměrům může být použit v racku nebo jako stolní jednotka.

Ovládání probíhá buď pomocí připojeného monitoru a klávesnice přímo do Simeny, nebo pomocí graficky zpracovaného webového rozhraní viz. obr.1.9. K přístupu na toto webové rozhraní musí být připojen PC pomocí monitorovacího portu k Simeně a Simena i PC musí být ve stejné síti.[8]



Obrázek 1.10: *Grafické rozhraní Simeny NE1000*

1.8 kvalitativní měření vytvořeného spoje

Hodnocení kvality vytvořeného spoje v telefonii nám poukazuje na to, jak kvalitně je telefonní trasa zbudována. Matematicky řečeno se jedná o zkoušku výsledku dané rovnice. Problematiky kvality paketového přenosu dat v telekomunikacích se týkají doporučení ITU-T G.107 a ITU-T G.108. Na měření kvality se můžeme dívat z několika úhlů. Můžeme měřit kvalitu vybudované trasy, kdy jsou měřené veličiny ovlivněny pouze fyzikálními parametry trasy a nás v takovém případě zajímají veličiny, jako jsou Jitter, BER, nebo zpoždění na trase. Nebo můžeme měřit kvalitu vytvořeného spojení na trase. Zde už musíme ovšem počítat i s chybami vzniklými použitými protokoly, kodeky a ostatními faktory potřebnými k vybudování telefonního spojení. Ve své práci budu měřit kvalitu vytvořené trasy pomocí síťového emulátoru Simena NE1000 popsaného výše. Na tomto síťovém emulátoru budu nastavovat parametry jako Jitter, ztrátovost a zpoždění paketů a sledovat, do kdy bude trasa schopná přenášet data, než se objeví první alarmy znázorňující rozpad spojení.

Popišme si tedy nejdříve, co která měřená veličina znamená.

1.8.1 BIT Error Rating

Bit Error Rating přeložíme do češtiny jako chybovost. Jedná se o bezrozměrnou veličinu, která vyjadřuje četnost chyb při přenosu. Je dána poměrem chybně přenesených bitů digitálního signálu, který ještě neprošel korekcí chyb k celkovému počtu přenesených bitů. Pokud budeme chtít přenést sekvenci bitů 0110110001 a na přijímací straně obdržíme 01_011_001, bude v tomto případě BER 2 – dva nesprávně přenesené bity rozděleny do 10 přenesených bitů. Výsledkem je tedy BER = 0,2 neboli 20%.

BER probability *pe* je pravděpodobná hodnota BER. Je to přibližný odhad počtu pravděpodobných chyb. Je dán počtem dekódovaných bitů, které zůstávají po nesprávné korekci chyb, dělený celkovým počtem bitů (dekódované užitečné informace). Čím delší je měřený časový interval a množství pravděpodobných chyb, tím je odhad přesnější.

(PER) – chybovost paketů, je počet chybně přijatých datových paketů dělený celkovým počtem přijatých paketů. Paket je prohlášen za chybový, pokud je alespoň jeden bit v paketu přenesen chybné. Očekávaná hodnota PER je označována jako *pp* (packet error probability), který pro délku rámce n bitů může být vyjádřen jako:

$$p_P = 1 - \left(1 - p_e\right)^N \tag{1.3}$$

za předpokladu, že bitové chyby jsou na sobě nezávislé. Pro malé pravděpodobnosti bitových chyb, to je přibližně:

$$p_p \approx p_e N \tag{1.4}$$

Podobné měření může být provedeno pro přenos rámců, bloků, nebo symbolů.

Na chybovost BER lze také pohlížet z matematického hlediska jako na pravděpodobnost výskytu daného jevu. Tuto problematiku rozebrali vědci Gilbert, Elliot nebo Bernoulli.

1.8.1.1 ***Bernoulliho model***

Jacob Bernoulli byl švýcarským matematikem a fyzikem, žijícím v období 1655 – 1705. Zajímal se o zákonitosti mocninné řady a významně přispěl k rozvoji teorie pravděpodobnosti. Pro nás je důležitý díky Bernoulliho schématu:

Provádíme sérii n nezávislých náhodných pokusů, ve kterých nastává sledovaný výsledek, náhodný jev A, s pravděpodobností P(A) = p, 0 < p < 1. Pravděpodobnost *Pn(k)* toho, že se v sérii vyskytne náhodný jev A právě k−krát, k = 0, 1, 2, . . . , n je rovna:

$$P_n(k) = \binom{n}{k} p^k \, (1-p)^{n-k} \quad , 0 \leq k \leq n. \tag{1.5}$$

V telekomunikacích užíváme tento model k procentuálnímu vyjádření ztrátovosti paketů při simulaci přenosu.

V Bernoulliho ztrátovém modelu je každá ztráta paketu náhodná, bez ohledu na to, zda byl předchozí paket ztracen, či ne. V tomto případě je zde pouze jeden parametr a to průměrná míra ztráty paketu *Ppl* , který může být matematicky popsán vzorcem:

$$P_{pl} = \frac{n_l}{n} \cdot 100 \qquad (1.6)$$

Kde n_l je množství ztracených paketů a n je celkové množství přenesených paketů.

V komunikačním řetězci může být chybovost na přijímací straně ovlivněna hlukem, rušením, zkreslením, problémy bitové synchronizace, útlumem. BER lze zlepšit výběrem silnějšího signálu (kvůli snížení přeslechů)nebo výběrem pomalejší a robustnější modulace nebo linkových kódů. Více o této problematice zde [9]

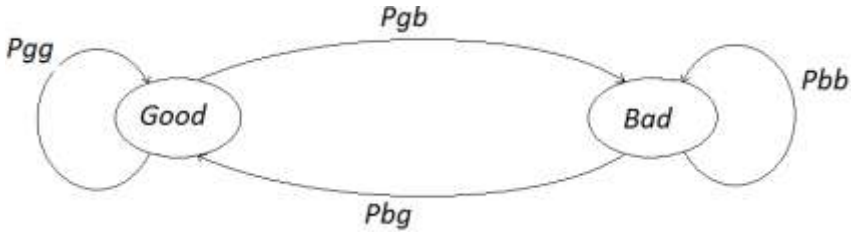
1.8.1.2 ***Gilbert-Eliotův model:***

Protože chybovost rámce je závislá na chování komunikačního kanálu, druhu použitého kódování, formátu rámců a například na časovém rozložení paketů, je potřebné simulovat rámce na bitové úrovni. Takovýto způsob simulace, aby byl přesný, vyžaduje minimálně jednu simulační událost pro každý bit. Díky Gilbert-Eliotova modelu se daří zachovat výsledky simulace na bitové úrovni i při simulaci na rámcové úrovni. Pro představu, jak Gilbert-Elliotův model funguje, můžeme uvažovat dva stavy – good a bad. Každý stav má přiřazenou určitou hodnotu BER. Pro dobrý stav je vyhrazena hodnota berg a špatnému stavu je přiřazena hodnota berb. Pro tyto dvě hodnoty platí rovnost berg << berb . Tím, že se chyby na přenosovém médiu vyskytují ve shlucích, je větší pravděpodobnost, že následující přijatý bit bude chybný, pokud právě přijatý bit byl také chybný. Označme proto pravděpodobnost, s jakou můžeme předpokládat, že po bezchybném bitovém příjmu bude i následující bit přijatý bezchybně jako Pgg (good-good). Pokud nastane situace, že bude po aktuálně přijatém bezchybném bitu, přijatý špatný bit, označíme tuto pravděpodobnost jako Pgb (good-bad), jako na obrázku 1.11.

Hodnoty Pbb (bad-bad) a Pbg (bad-good) můžeme odvodit analogicky. Platí:

$$P_{gg} = 1 - P_{gb} \qquad \text{a taktéž} \qquad P_{bb} = 1 - P_{bg} \qquad (1.7)$$

Pro výslednou simulaci je důležité uchovat informaci o tom, v jakém stavu skončilo přijetí posledního rámce, protože to ovlivní simulační chybovost následujícího rámce. Možnstí, jak upřesnit základní variantu Gilbert-Elliotova modelu je, že nadefinujeme vyšší počet stavů, například 2 good a 2 bad stavy. Simulace však ukázaly, že vyšší počet stavů než 4 už téměř neovlivňuje dosažené výsledky. [10]



Obrázek 1.11: *Gilbert-Elliotův model*

1.8.2 **Jitter**

Dalším parametrem, kterým proměříme kvalitu trasy je Jitter.

Tento parametr přenosové trasy vyjadřuje nežádoucí kolísání (nestabilitu) zpoždění a je taktéž jako zpoždění vyjadřován v milisekundách. Jitter v systémech PDH 2.048 kbit/s je posán v doporučení ITU-T G.823. Stejně tak, jako tomu je u zpoždění, jsou i na vyšší hodnoty parametru jitter služby internetu velmi náchylné (např. již výše uvedené telefonování po internetu). V praxi se toto zpoždění projevuje tak, že pokud porovnáme rozdíly mezi datagramy vyslanými na straně odesilatele a poté tyto rozdíly porovnáme s rozdíly mezi datagramy přijatými na straně příjemce, uvidíme, že nejsou totožné. Pakety totiž neměly stejné podmínky při přenosu, protože strávily odlišný čas ve frontách na směrovačích. Tímto tedy došlo k rozdílu mezi skutečným a očekávaným časem příchodu. Tento rozdíl je pak označen jako jitter (rozptyl). Podle ITU-T G.823 je maximální povolená hodnota Jitteru v pásmu 20Hz až 100KHz 1,5 UI. Jednotka UI znamená jednotkový interval (Unit Interval), který má při přenosové rychlosti 2048 Kbit/s hodnotu 488 ns. Maximální povolená hodnota je tedy 732 ns. K vykompenzování Jitteru může být použit Jitter buffer.[11]

1.8.3 **Zpoždění**

Zpoždění, neboli latence vyjadřuje v telekomunikacích zpoždění signálu na přenosové trase. Problém latence je rozložen do celé délky přenosové trasy. Jedná se o čas, který uplyne od odeslání signálu zdrojovým uzlem po jeho přijetí na cílovém uzlu. Zahrnuje zpoždění na přenosové trase a taky na zařízeních, které jsou součástí této trasy. Je nutné rozlišit jednosměrné zpoždění (čas mezi odesláním paketu od zdroje a jeho přijetím na přijímací straně) a zpoždění round-trip latency, v překladu znamenající obousměrné. Toto zpoždění zahrnuje dobu cesty paketu tam i zpět plus čas jeho zpracování v cíli. Round-trip latency nebo také Round Trip Time (RTT) se v síťové praxi požívá nejčastěji, protože jej lze změřit z jednoho místa (uzlu). V rámci pobočkové ústředny PBX se doba přenosu signálů pohybuje zpravidla pod 10 milisekund, v rámci širší oblasti nebo regionu se doba přenosu signálů pohybuje zpravidla pod 50 milisekund. Tato doba je díky nedokonalosti lidského ucha nevnímatelná a tudíž i dostačující pro přenos hlasu. Problémy ale mohou vznikat při mezinárodních a

mezikontinentálních hovorech (např. při využití družicových spojovacích systémů), kde doba přenosu zpráv může být až 400 milisekund, což je doba, kterou lidské ucho i přes svou nedokonalost postřehne a hovor se v tu chvíli stává obtížným, protože obě strany mají pocit, že je ta druhá neslyšela a začínají si skákat do řeči. Latence je tedy velmi důležitá pro kvalitu hlasové komunikace a v rámci telefonní sítě nemá překračovat hodnotu 300 milisekund včetně koncového zařízení.

2 Projekt Asterisk

Projekt Asterisk byl zahájen v roku 1999, kdy Mark Spencer vydal první kód pod GPL open-source licencí. Dnes je Asterisk udržován za společného úsilí společnosti Digium Asterisk a komunitou uživatelů.

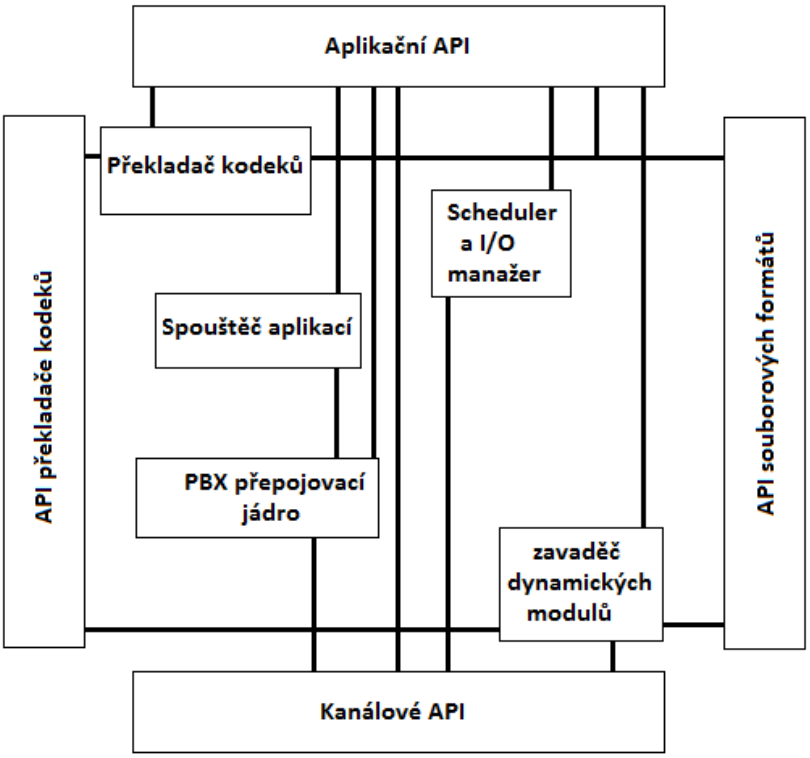
Na otázku, co je to Asterisk se dá odpovědět mnoha způsoby. Například, že se jedná o open-source software, což znamená možnost komukoli upravovat zdrojový kód podle svých vlastních potřeb a vydávat nová rozšíření, které je taky možno měnit. Z toho také vyplývá, že program je možno stáhnout bezplatně z oficiálních stránek vývojáře.

Jiným způsobem lze odpovědět, že se jedná o symbol hvězdičky (*), který je obsažen i v logu tohoto softwaru. Hvězdička je v v počítačové populaci chápána jako symbol pro libovolný znak. Vývojáři tak chtěli poukázat, že hlavní myšlenkou při vývoji tohoto softwaru bylo vymyslet komplexní řešení splňující všechny telefonní požadavky.

Ovšem software Asterisk je systém, který nám umožňuje vytvořit softwarovou pobočkovou ústřednu pro uskutečnění hovorů jak IP, tak i digitální ISDN a také analogově dle našich požadavků. Navíc nám stačí klasické PC, není třeba dalších hardwarových součástek pro VoIP, ačkoli je třeba implementovat nestandardní ovladač, který nahradí fiktivní hardware potřebný u některých aplikací (například konferenční hovory). Program je vyvíjen pro Linux a Unix, nicméně po emulaci běží i na ostatních systémech (BSD, Windows, OS X). Podporuje VoIP ve čtyřech protokolech a může spolupracovat s téměř všemi telefonními standarty při využití relativné levného hardware. Podporuje hlasovou schránku spolu s adresářem, konferenční volání nebo služby ID volajícího a další. Mezi podporované protokoly patří protokoly ADSI, SIP, H.323 (klient i brána). S nimi pracuje jako s kanály připojenými k jádru systému. Srdcem systému je Dialplan, kde jsou nadefinovány veškeré postupy v případě příchozího nebo odchozího požadavku. Ovládání je zajištěno skrze příkazový řádek CLI nebo Manager Interface. Další výhodou je možnost obsluhy příchozích či odchozích hovorů různými VoIP poskytovateli.

2.1 Architektura Asterisk

Asterisk je koncipován tak, aby bylo možné využívat každý vhodný hardware a technologii (i do budoucna) za účelem propojování HW a aplikací. Viz obrázek 2.1. Skládá se z jádra systému a okolních API. Jádro systému ovládá specifické protokoly, kodeky a HW rozhraní telefonních aplikací, zatímco okolní API slouží pro usnadnění oddělení protokolů a hardware, proto jádro ústředny Asterisk nemusí řešit, jaké kodeky používá volající nebo jak se připojí.



Obrázek 2.1: *Architektura Asterisku*

PBX spojování (PBX Switching), Primárním cílem Asterisku je bezpochyby propojování v PBX, spojování mezi uživateli a automatizovanými úlohami. Přepojovací jádro slouží k transparentnímu spojování příchozích volání na různých HW a SW rozhraních.

Spouštěč aplikací (Application Launcher) dovoluje spouštět aplikace zajišťující služby, jako jsou kupříkladu hlasová pošta, výpis adresáře a přehrání souboru.

Překladač kodeků (Codec Translator) užívá moduly kodeků ke kódování a dekódování zvukových kompresních formátů používaných při telefonii. Velké dostupné množství kodeků je vhodné pro rozmanité potřeby a docílení stavu rovnováhy mezi kvalitou zvuku a použitou šířkou pásma.

Scheduler a I/O manažer, v originále (Schedule and I/O manager) je k ovládání nízkoúrovňových úloh a systémového řízení aby byl zajištěn optimální výkon podle stavu zatížení.

Kanálové API slouží k ovládání typu spojení příchozího volání, tedy jedná-li se o spojení VoIP, ISDN PRI nebo jakoukoli jinou další technologii. Dynamické jednotky jsou zavedené k ovládání detailů nižších vrstev v těchto spojeních.

Aplikační API bere v úvahu libovolné úlohy, které mají běžet tak aby realizace funkcí jako hlasová pošta, paging, konference a každý následující úkol, který by systém PBX mohl vykonávat (teď nebo dobudoucna), byla ovládána prostřednictvím těchto oddělených modulů.

API překladače kodeků zavádí moduly kodeků k podpoře různých audio formátů kódování nebo dekódování jako GSM, A – law, μ - law a nebo MP3.

API souborových formátů slouží k ovládání čtení a zápisu různých souborových formátů určených k ukládání dat v souborovém systému. Více o Asterisku se lze dočíst zde[12]

2.2 Instalace Asterisku a závislostí

Distribuce Asterisk je schopna fungovat na všech distribucích Linuxu, ale primárně byla již od svého počátku vyvíjena pro distribuci CentOS. Proto jsem se rozhodl nainstalovat CentOS ve verzi 5.1 abych pak mohl použít Asterisk 1.8, který je nejprobíranější na fórech a tudíž, je k němu nejvíce podkladů. Nicméně toto řešení s sebou přineslo více problémů, než kladů. Pro verzi jádra 2.6.18 nebylo možné stáhnout doplňkové knihovny DAHDI v kompletní verzi DAHDI-Linux-complete a bylo nutné hledat jednotlivé balíčky zvlášť. Další překážkou byla nemožnost komunikace na tcp protokolu. proto jsem se rozhodl od této distribuce upustit a přejít k distribuci Linux mint. Po jeho instalaci je nutné provést aktualizaci na nejnovější verzi systému. Tu provedeme zadáním příkazu *#apt-get update*, po instalaci zadáme *#apt-get upgrade* a opět vyčkáme, než se instalace dokončí a poté restartujeme celý systém příkazem #restart. Dalším nezbytným krokem je instalace balíčku ntp, jenž slouží k udržování přesného času kvůli synchronizace Asterisku se zbytkem světa. Zadáme tedy příkaz *#apt-get install ntp* a odsouhlasíme stiskem *#y*. Ještě před samotnou instalací asterisku je třeba nainstalovat softwarové závislosti požadované Asteriskem. Tyto balíčky nám umožní vytvořit základní Asterisk systém, spolu s DAHDI a LibPRI. Pokračujeme tedy příkazy *#apt-get install build-essential subversion, #apt-get install libncurses5-dev libssl-dev libxml2-dev libsqlite3-dev a #apt-get install uuid-dev vim-nox*. Tyto balíčky nám dají většinu toho, co budeme potřebovat k instalaci Asterisku, DAHDI a LibPRI.

Když máme nainstalovány závislosti, vytvoříme si adresářovou strukturu, kam budeme stahovat a instalovat stažené balíčky. Složky asterisk-complete a asterisk vytvoříme najednou příkazem *#mkdir –p ~/src/asterisk-complete/asterisk*. Nyní můžeme přistoupit ke stažení a instalaci balíčků se striktně dodrženým pořadím: DAHDI, LibPRI, Asterisk 1.8. Zachovat toto pořadí je důležité kvůli zajištění nainstalování všech závislostí pro DAHDI a Asterisk ještě před spuštěním konfiguračních skriptů, které zajistí postavení všech modulů závislých na LibPRI nebo DAHDI.

2.2.1 Instalace DAHDI

DAHDI je zkratka pro „Digium Asterisk Hardware Device Interface“ a jedná se o open-source software Asterisk určený pro komunikaci a ovládání telefonního hardwaru, například karet Digium.

Ovšem ne vždy se jednalo o DAHDI. V předchozích verzích, pro přidání TDM podpory do Asterisku začala vyvíjet firma ZAPATA Telephony pseudo TDM rozhraní, které nazvala Zaptel. Toto jméno bylo dle některých zdrojů inspirováno mexickým revolucionářem generálem Emiliano Zapatou, protože stejně jako on, se i firma Zapata rozhodla provést razantní, až revolucionářské změny v oblasti telefonie. Pseudo TDM architektura poskytuje téměř stejnou kvalitu a real-time efektivitu, jakou má hardware TDM. Nespornou výhodou je však podstatně nižší cena a obrovská flexibilita. Zaptel a později i DAHDI rozhraní dodává firma Digium.

DAHDI je seskupení dvou samostatných utilit DAHDI-Linux a DAHDI Tools. DAHDI Linux poskytuje ovladače jádra a DAHDI Tools nám zase poskytuje různé nástroje pro správce jako je dahdi_cfg nebo dahdi_scan a jiné.

Pro instalaci DAHDI je důležité vědět, jakou máme nainstalovanou verzi jádra Linux, protože je třeba vybírat tutéž verzi pro DAHDI. Nebo necháme systém, aby si verzi zjistil sám a podle ní stáhl odpovídající verzi DAHDI. Instalaci provedeme pomocí těchto příkazů:

```
#cd ~/src/asterisk-complete/dahdi

# wget http://downloads.asterisk.org/pub/telephony/dahdi-linux-
complete/dahdi-linux-complete-current.tar.gz

# tar –zxvf dahdi-linux-complete-current.tar.gz

# cd dahdi-linux-complete-current

# make all

# make install

#make config
```

Tímto jsme úspěšně nainstalovali DAHDI. Dalším v pořadí je instalace LibPRI.

2.2.2 Instalace LibPRI

LibPRI je knihovna, která přidává podporu pro ISDN (PRI a BRI). Pro naši konfiguraci jsme užili verze 1.4.3.

```
#cd ~/src/asterisk-complete

#mkdir libpri

#cd libpri/

# wget
http://downloads.asterisk.org/pub/telephony/libpri/libpri-1.4-
current.tar.gz

#tar –zxvf libpri-1.4-current.tar.gz

#cd libpri-1.4-current
```

```
#make
#make install
```

Tímto jsme nainstalovali LibPRI. Po instalaci balíčků LibPRI a DAHDI máme všechny potřebné závislosti nainstalovány a můžeme přistoupit k instalaci samotného programu Asterisk. Proto stáhneme verzi 1.8 ze stránek výrobce a již známými příkazy make a make install nainstalujeme.

2.2.3 Instalace Asterisk 1.8

```
#cd ~/src/asterisk-complete/asterisk
#wget
http://downloads.asterisk.org/pub/telephony/asterisk/asterisk-
1.8-current.tar.gz
#tar zxvf asterisk-1.8-current.tar.gz
#cd asterisk
#./configure
#make
#sudo make install
#sudo make config
```

Striktní dodržení tohoto návodu zajistí bezproblémový chod softwarové pobočkové ústředny Asterisk. na osobním počítači. Pro vytvoření trasy pro přenos TDM přes Ethernet je ovšem třeba použít dvě karty Digium TE110P a tudíž i dva osobní počítače. Proto je třeba celý tento postup aplikovat ještě jednou i pro druhý počítač.O Asterisku a jeho konfiguraci pojednává elektronická kniha dostupná zde [4]

2.3 Telefonní karty

Pro připojení telefonů k systému Asterisk je potřeba použít telefonní kartu. Takováto telefonní karta není ovšem záležitostí jedné firmy a i v tomto odvětví je konkurence vysoká. Pro příklad uvedu tři nejznámější výrobce telefonních karet, mezi které patří OpenVox, Sangoma nebo Digium.

Karty OpenVox nabízí většinu pokročilých funkcí pro VoIP a jsou kompatibilní s open-source systémy nebo ostatním OpenVox hardwarem. Firma nabízí mnoho typů karet pro analogové, BRI, PRI, GSM nebo E1 nebo T1 rozhraní. Pro praktickou část bakalářské práce je možné použít od tohoto výrobce kartu DE130P. Více na stránkách výrobce [13]

Dalším výrobcem karet je, jak již bylo zmíněno, Sangoma. Tato společnost byla založena roku 1984 a zaměřuje se na poskytování hardwarových a softwarových komponent pro IP komunikaci. Její telefonní karty jsou již od výroby plně kompatibilní s open-source systémy jako Asterisk nebo FreeSWITCH. Karty pracují na rozhraních analog, BRI,PRI, E1, T1, J1 nebo také GSM. Pro praktickou část mé bakalářské práce by bylo možné použít kartu A101 od tohoto výrobce. Oficiální stránky výrobce zde [14] Další firmou je firma Digium, založená vývojářem Asterisku Markem Spencerem. Karty distribuované touto firmou jsou stvořené přímo pro Asterisk a dodávají pro tento software nejlepší podporu. Po konzultaci s vedoucím mé práce mi bylo doporučeno použít kartu TE110P právě tohoto výrobce. Více informací o společnosti Digium a jejích výrobcích zde [15]

2.4 Digium TE110P

TE110P je single span karta zobrazená na obrázku 2.2, na které můžeme používat jak E1 (32 kanálů) používané v Evropě, tak T1 (24 kanálů) používané v USA. Toto přepínání kanálů se provádí pomocí Jumperu umístěného přímo na kartě. Pokud se Jumper nachází v pozici ON, je nastaveno E1 a pokud je Jumper v pozici OFF, je nastaveno T1. Karta podporuje všechny funkce T1/E1 karty Digium. Tato karta podporuje i hlasové a datové režimy na svém jediném rozpětí. Například, karta může podporovat 12 hlasových a 12 datových kanálů při průchodu veškerého provozu až do Asterisk PBX, která spolehlivě nasměruje kanály do určených míst. To eliminuje potřebu externího routeru.

Využitím TDMoE technologie, zprostředkované firmou Didium, lze snadno připojit více počítačů vybavených touto kartou a dosáhnout tak přenosu na úrovni hlasové kvality.Výhodou tohoto produktu je přidávání libovolného počtu karet ke každému jednotlivému PC. Karta podporuje standardizované průmyslové telefonii a datové protokoly, včetně RBS a Primary Rate ISDN (PRI) pro hlas a PPP, Cisco HDLC, a Frame Relay pro režimy dat. Karta je kompaktní a výkonné rozhraní podporující přenos hlasu a dat přes T1, E1, a Primary Rate ISDN (PRI) připojení.

Tato karta se nejčastěji používá, jak už název napovídá, ve spojení s Asterisk open source systémem. Spojením těchto dvou technologií lze vytvořit bezproblémové sítě, propojení PSTN s Voice-over IP.

Obrázek 2.2: *karta Digium TE110P*

Jak je vidět na obrázku karta Digium TE110P karta se připojuje do PCI slotu. Po jejím připojení a znovu zapnutí počítače je patrné, že karta sice připojená je, ale s počítačem nekomunikuje. To lze napravit párem jednoduchých příkazů. Nejprve je třeba detekovat nově přidaný hardware (Digium TE110P) příkazem

```
# dahdi_genconf
```

Tento příkaz načte kartu do systému a dalším příkazem nastavíme kartu pro základní použití.

Zadáme proto příkaz

```
# dahdi_cfg-vv
```

Tímto je vytvořen a nastaven pro základní použití konfigurační soubor pro Digium kartu system.conf ve složce /etc/dahdi/.

Pro korektní využití karty se softwarem Asterisk je třeba ještě restartovat DAHDI a Asterisk v tomto pořadí příkazy:

```
# / etc / init.d / dahdi restart
# / etc / init.d / asterisk restart
```

Správnost postupu lze ověřit jak blikáním červené LED na kartě, tak v systému Asterisk. Zadáme proto příkaz ke spuštění Asterisku

```
#asterisk –rvvvvvv
```

a napíšeme

```
*CLI>dahdi show status
```

a měli bychom vidět takovýto stav:

Tabulka 2.1: *tabulka zobrazující výstup po zadání dahdi show status*

Description	Alarms	IRQ	bpviol	CRC4
Digium Wildcard TE110P T1/E1 Card 0	OK	0	0	

Stejně tak lze ověřit nastavení kanálů karty, to by mělo vypadat takto pro všech 31 kanálů:

```
asterisk*CLI> dahdi show channels
```

Tabulka 2.2: *Tabulka zobrazující výstup po zadání dahdi show channels*

Chan Extension	Context	Language	MOH interpret
pseudo	default		In service
1	from-pstn		In service
2	from-pstn		In service
3	from-pstn		In service
:	:		:
:	:		:
31	from-pstn		In service

[16]

2.5 Konfigurace E1

pro konfiguraci E1 v systému Asterisk je nutné upravit konfigurační soubory system.conf ve složce /etc/dahdi/, soubor chan_dahdi.conf ve složce /etc/asterisk/ a soubor modules.conf opět ve složce /etc/asterisk/. Nejprve upravíme soubor system.conf do podoby níže:

```
# Span 1: WCT1/0 "Digium Wildcard TE110P T1/E1 Card 0" (MASTER)
CCS/HDB3/CRC4

span=1,1,0,ccs,hdb3,crc4

# termtype: te

bchan=1-15,17-31

dchan=16

echocanceller=mg2,1-15,17-31


# Global data

loadzone    = cz

defaultzone = cz
```

V tomto nastavení vidíme v prvním řádku název použité karty a shrnutí jejího nastavení. V druhém řádku je nastaven rozsah karty ve formátu

span=<span num>,<timing source>,<line build out (LBO)>,<framing>,<coding>

První číslo udává pořadí karty a není třeba se jím více zabývat pokud máme osazenu pouze jednu kartu na jednom PC. Druhé číslo znázorňuje generování hodinového signálu. Tento parametr určuje, zda bude hodinový signál ze vzdáleného konce E1 považován za zdroj časování. Pokud zvolíme 1, bude zařízení na opačné straně považováno za master zdroj hodinového signálu. Číslo 2 se zde nastaví, pokud chceme toto zařízení nastavit jako druhou volbu zdroje hodinového signálu pokud první zdroj selže. Taktéž pro 3 atd. Pokud nastavíme 0, port nebude nikdy použit jako zdroj hodinového signálu. V pořadí třetí číslo označuje <line build out (LBO)> a vybírá se zde z tabulky 2.3 níže

Tabulka 2.3: *tabulka LBO*

0	0 db (CSU) / 0-133 feet (DSX-1)
1	133-266 feet (DSX-1)
2	266-399 feet (DSX-1)
3	399-533 feet (DSX-1)
4	533-655 feet (DSX-1)
5	-7.5db (CSU)
6	-15db (CSU)
7	-22.5db (CSU)

Pro framing nastavíme centralizovanou signalizaci ccs (Common Channel Signalling) používanou pro E1. pro kódování nastavíme kód HDB3 používaný také pro E1.

V řádku bchan nastavujeme definování používání kanálů. Totéž platí i o dchan, kde nastavíme kanál pro signalizaci.

v řádku echocanceller nastavujeme potlačení ozvěny pro každý kanál. V základním nastavení jsou tyto potlačení ozvěny zakázány, proto je třeba je povolit v tomto řádku, který má syntaxi echocanceler = <echocanceller name>, <channel(s)>

my použijeme pro všechny datové kanály potlačení ozvěny mg2.

Posledním nastavením v tomto konfiguračním souboru je nastavení použitých tónů pto danou zeměpisnou oblast ve které se nacházíme. Formát tohoto nastavení je loadzone=<zone> a defaultzone=<zone>, kde za zone dosadíme písmenný kód země, pro Českou republiku tedy zapíšeme cz.

Dalším souborem, který je nutné nastavit je konfigurační soubor chan_dahdi.conf uložen ve složce /etc/asterisk/. Pokud zde není, opět jej vytvoříme například pomocí textového editoru nano a upravíme do této podoby:

```
[channels]
group = 11
;echocancel = yes
signalling = pri_cpe
switchtype = euroisdn
channel => 1-15,17-31
context = default
```

Slovo group označuje volací skupinu. Číslo této skupiny se volí v rozsahu 0-63 a pro chod linky není důvod jej měnit z defaultního nastavení. Obvykle se volí jedna skupiny pro všechny PRI kanály.

Asi nejdůležitější je zde nastavení signalling. To se nastavuje jako pri_cpe signalizace pro PRI Slave stranu a pri_net pro PRI Master stranu.

switchtype se používá pouze pro ISDN PRI připojení a pro použití v Evropě jej nastavíme na hodnotu euroisdn.

V řádku channels volíme kanály používané pro přenos dat.

context značí užitý kontext pro volání, užitý v souborech sip.conf a extensions.conf. Defaultně je tento kontext nastaven na default a není nezbytně nutné jej upravovat.

Jako poslední zmiňovaný konfigurační soubor upravíme modules.conf. Tento soubor Asterisku umožňuje hledat a ovládat jeho dodatečné moduly, které najde ve složce *usr/lib/asterisk/modules*.

V tomto souboru, jak už je u Asterisku zvykem, můžeme definovat mnoho nastavení. Ovšem pro jednoduchost a čistotu řešení je nejlepší umožnit Asterisku načítat zde všechny

moduly které najde a nezabývat se vyčleňováním modulů které nepoužijeme. Upravíme proto tento soubor následovně:

```
[modules]
autoload=yes
```

Tímto jsme nastavili E1 pomocí dvou PC se softwarem Asterisk a dvěma kartama Digium TE110P, osazené každou do jednoho PC. Funkčnost tohoto řešení můžeme ověřit uskutečněním hovoru. Proto si vytvoříme na každém PC SIP účet a upravíme soubor extensions.conf kvůli nastavení dialplanu.

2.6 Nastavení SIP účtů

Abychom mohli vůbec uskutečnit hovor, je třeba se nyní zaobírat nastavením uživatelských účtů. Nejprve je nutné zvolit si protokol, který budeme využívat ke směrování hovorů, a od toho se bude odvíjet další konfigurace. V tomto ohledu Asterisk nabízí mnoho možností, kupříkladu můžeme využít protokoly IAX2 a SIP. K nastavení těchto protokolů se využívá souborů */etc/asterisk/sip.conf* pro SIP anebo */etc/asterisk/iax.conf* pro zařízení pracující s protokolem IAX2. Tyto dva jsou nejpopulárnější protokoly pro propojení koncových zařízení. Pro náš účel použijeme protokol SIP.

Otevřeme si v našem oblíbeném textovém editoru soubor *sip.conf* a vytvoříme si zde SIP účet.

Na jednom PC jsme si vytvořili účet s názvem 201.

```
[201]
type=friend
context=default
callerid=201
secret=heslo
host=dynamic
dtmfmode=rfc2833
disallow=all
allow=alaw
```

Na druhém PC jsme si také vytvořili SIP účet, ovšem s názvem 200.

```
[200]
type=friend
```

```
context=default
callerid=200
secret=heslo
host=dynamic
dtmfmode=rfc2833
disallow=all
allow=alaw
```

Vysvětleme si nyní, co toto nastavení znamená.

Jméno zařízení je uvedeno v hranatých závorkách. Tento název je libovolný a uživatel si může zvolit jakékoliv jméno.

Type=friend Znamená, že ovladač kanálu se bude pokoušet najít nejprve jméno a až potom IP adresu. (Peer = IP adresa + port. User = hledá název v SIP hlavičce „From“, který se shoduje se jménem v hranatých závorkách).

context nastaví jakým kontextem v dialplanu bude obsloužen tento uživatel.

callerid udává název pobočky, který se zobrazí na displeji telefonu.

Secret=****** nastaví heslo pro tento účet (heslo je to, co je za rovnítkem).

host=dynamic udává užití dynamické adresy, kterou předá Asterisku, jinak je možné napsat statickou IP.

dtmfmode určuje generování tónů

Disallow=all zakáže všechny dosud povolené kodeky.

Allow=alaw,ulaw,gsm naopak povolí uvedené kodeky.

Po uložení nastavené konfigurace je třeba ji načíst do systému *chan_sip.so*. Zadáme proto

```
#asterisk –r
*CLI> module reload chan_sip.so
```

O správnosti konfigurace se můžeme přesvědčit zadáním

```
*CLI> sip show peers
```

nebo

```
*CLI> sip show users
```

2.7 Extensions.conf

Dalším krokem ke konfiguraci SIP účtů je nastavení Dialplanu v souboru extensions.conf. Tento soubor nalezneme v /etc/asterisk.

Dialplan je jakýmsi srdcem Asterisku. Říká, jak budou hovory vstupovat ze systému a do něj. Vytváří se pomocí skriptovacího jazyka, pomocí kterého definujeme instrukce, které Asterisk provádí jako odpověď na podněty z vnější. Oproti tradičním telefonním systémům je dialplan možné si plně přizpůsobit. Dělí do 4 základních koncepcí. Na kontexty, priority, pobočky a aplikace.

Kontexty jsou uváděny v hranatých závorkách a oddělují části dialplanu od sebe. Pobočka a pravidla pro ni nastavená v jednom kontextu, jsou naprosto izolována od pobočky z kontextu druhého. To ale můžeme povolit, pokud chceme použitím takzvaných šablon. Dialplan má již od počátku naimplementovány dva existující speciální kontexty nesoucí názvy [general] a [globals]. První kontext obsahuje seznam obecných nastavení pro dialplan a druhý kontext obsahuje globální proměnné. Kromě těchto dvou kontextů by bylo vhodné vyvarovat se používání kontextu [default].

V Asterisku definuje pobočka pod daným názvem jedinečnou posloupnost kroků, přes které Asterisk hovor povede. Uvnitř každého kontextu máme možnost definovat neomezený počet poboček. Syntaxe pro telefonní pobočku tvoříme slovem exten => , za kterým následuje název nebo číslo pobočky nebo kombinace. Poté následuje priorita a za ní je oddělena čárkou aplikace, což je příkaz, který se má v daném kroku provést.

```
exten => název, priorita, aplikace()
```

Priority jsou číslovány popořadě od čísla 1 a každý krok spustí specifickou aplikaci. Toto číslování nemusí být striktně dodrženo ve všech krocích, můžeme také nastavit automatické doplnění čísla s prioritou vyšší o 1 než je předchozí krok a to nahrazením daného čísla priority písmenem n. První priorita však musí začínat číslem 1 a až poté můžeme používat toto nahrazení i několikrát za sebou.

Každá aplikace dělá specifickou akci s daným kanálem, ku příkladu přehrání zvuku, čtení DTMF volby, volání kanálu, prohledávání databáze, zavěšení hovoru a podobně. Příklady aplikací jsou Answer(), Playback(), Progress() a tak dále.

Pro naše nastavení bude extensions.conf na počítači se SIP účtem 201 vypadat takto:

```
[default]
exten => 888,1,Dial(DAHDI/G11/${EXTEN})
exten => 200,1,Dial(DAHDI/G11/${EXTEN})
```

```
exten => 201,1,Dial(SIP/201)


exten => 555,1,Answer()
exten => 555,n,Playback(hello-world)
exten => 555,n,HangUp()
```

a na počítači s účtem 200 bude jeho nastavení obdobné, pouze pro jiná čísla.

```
[default]
exten => 201,1,Dial(DAHDI/G11/${EXTEN})
exten => 555,1,Dial(DAHDI/G11/${EXTEN})

exten => 200,1,Dial(SIP/200)


exten => 888,1,Answer()
exten => 888,n,Playback(hello-world)
exten => 888,n,HangUp()
```

Jak je vidět, oba kontexty jsou opticky rozděleny na tři části, kdy pokaždé první část umožňuje volat skrze DAHDI testovací číslo protistrany a protistranu samotnou. Druhá část definuje SIP uživatele a třetí část umožňuje volat ústřednu, která přehraje automatickou zprávu a zavěsí.

Kontext s názvem default již máme svázaný s uživateli 200 a 201 v konfiguračním souboru /etc/asterisk/sip.conf, takže nyní stačí pouze načíst novou konfiguraci do Asterisku. Po uložení konfigurace ji načteme do Asterisku reloadem celé konfigurace CLI asterisku zadáním příkazu

```
#dialplan reload
```

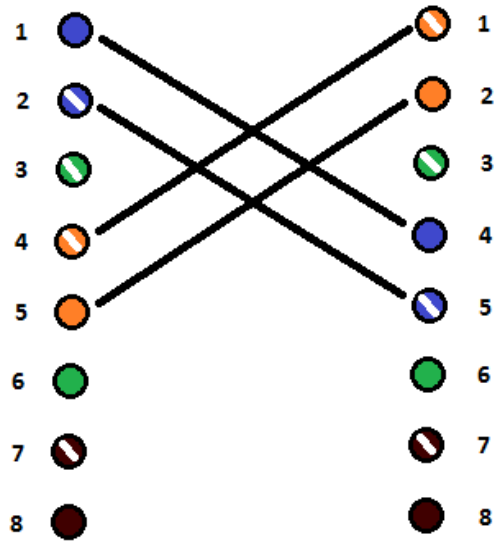
Více o Projektu Asterisk a nastavení SIP účtů se lze dočíst na webu wiki.asterisk dostupného zde **Chyba! Nenalezen zdroj odkazů.**

2.8 Vytvoření kabelu

Abychom mohli propojit naše dvě karty Digium TE110P, je třeba vytvořit kabel s konektory RJ45 pro E1. K tomu použijeme UTP kabel cat5 a jednotlivé žíly zapojíme

v pořadí 1,2 a 4,5, což musí býti pochopitelně do kříže (čili 1,2 z jednoho konektoru RJ45 půjde na 4,5 druhého konektoru RJ45 a naopak) jak je vidět na obrázku 2.3.

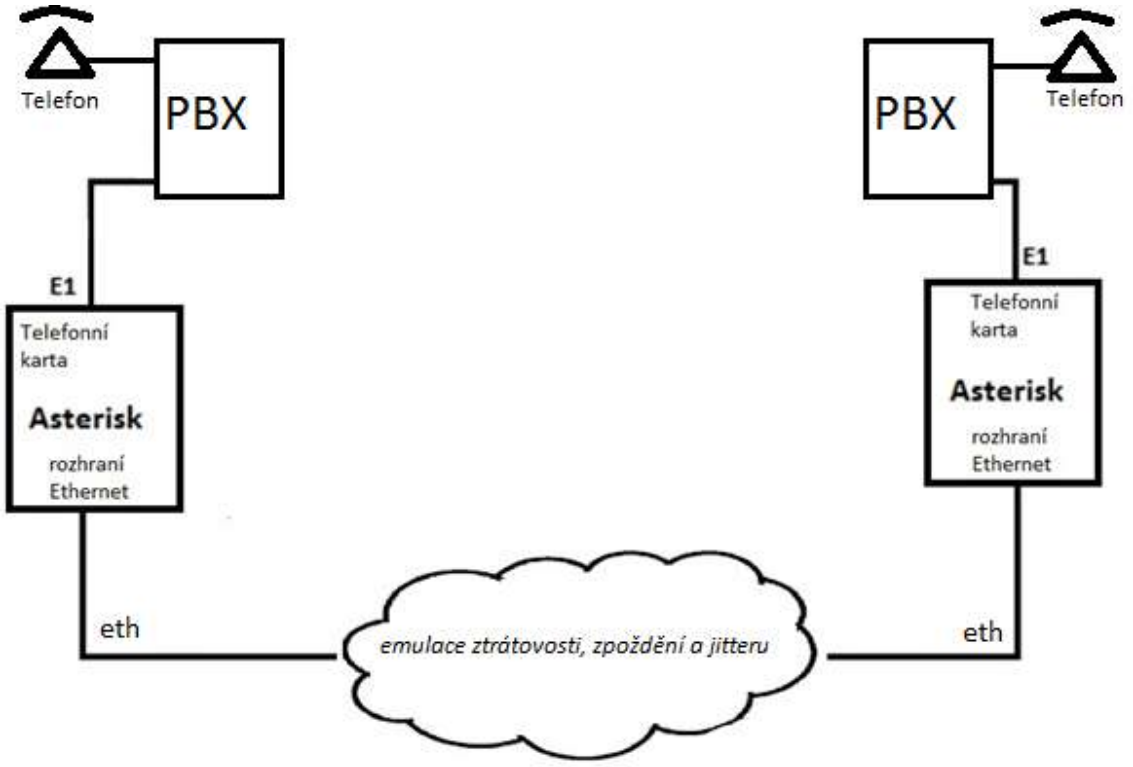
.



Obrázek 2.3: *umístění žil*

3 Praktická realizace E1 tunelované přes Ethernet

V této kapitole se zabývám praktickou konstrukcí zapojení v laboratoři, na kterém budu zkoumat kvalitu a limitní podmínky pro přenos. Konstruoval jsem zapojení dle obrázku 3.1, za použití dvou osobních počítačů s operačním systémem Linux Mint, na které jsem nainstaloval Asterisk 1.8 a každý z těchto počítačů osadil jednou kartou Digium TE110P. Dále jsem si vytvořil propojovací kabel s koncovkami RJ45 a jím tyto karty propojil. K tunelování jsem využil ethernetového rozhraní v PC.



Obrázek 3.1: *E1tunelovaná přes Ethernet*

3.1 TDMoE

Ze všeho nejdříve si vysvětleme tuto zkratku. Jedná se o spojení dvou anglických zkratek TDM a E, kdy TDM znamená Time-division-multiplex a E je zkratkou pro Ethernet.

Time-division-multiplex neboli multiplex s časovým dělením jsme si vysvětlili v předchozí kapitole.

Ethernet je nejrozšířenější technologie pro budování počítačových sítí. Hlavní příčinou úspěchu Ethernetu je jeho jednoduchost a nízká cena. Pojem Ethernet obvykle zahrnuje různé varianty této technologie – Fast Ethernet, Gigabit Ethernet, 10G Ethernet, 40G Ethernet a 100G Ethernet. Jako přenosové médium se nejčastěji využívá kroucená dvojlinka (běžná přenosová

rychlost 100 nebo 1000 Mbps), nebo dříve často užíván koaxiální kabel (10 Mbps). V užším smyslu je Ethernet síťový standard pro přenos dat rychlostí 10 Mbit/s, který byl vytvořen koncem 70. Let. Ethernet je realizován pomocí jednoduché přístupové kolizní metody CSMA/CD, kdy stanice vysílá data, kdy chce a pokud se sejdou dvě stanice vysílající různá data ve stejnou chvíli, vygeneruje se kolizní signál JAM a všechny stanice, které v daném okamžiku vysílaly, generují náhodný časový interval, po kterém se pokusí vysílat znova.

TDMoE protokol je využíván v situacích, kdy je potřeba spolehlivosti časového multiplexu bez jeho tradičního hardwaru. V počítačové technice Můžeme přenést hlas tak, že jej digitalizujeme, poté rozdělíme na jednotlivé pakety a rozešleme sítí. Síťové pakety mohou přenášet data velmi flexibilně, protože každý proud je rozdělen na malé kousky, kde je každý posílán s vlastním záhlavím popisujícím jejich obsah. To s sebou ovšem stále neslo několik problémů (Pakety mohou být poškozené, je potřeba mnoho prostoru pro záhlaví paketu, neexistuje záruka spolehlivosti). Kvůli zefektivnění přenosu byl vynalezen TDM (Time Division Multiplexing).

TDM stále posílá digitální data v blocích, ale s velmi malou hlavičkou (jen pro označení začátku paketů) a pak jsou data zřetězeny ve specifickém pořadí. To znamená TDM, protože kanály nejsou rozlišovány podle záhlaví, ale podle času kdy dorazí na demultiplex. Tohle je pro telefonii mnohem přirozenější, protože je stále vše velmi jednoduché, a přesto je zachována myšlenka obvodu, kde jsou digitální linky ještě rozděleny do určitého počtu kanálů, mezi kterými nepřepínat podobně jako na telefonní lince. TDMoE je užitečné, protože umožňuje vyšší flexibilitu a spolehlivost časového multiplexu přes levný Ethernet síť namísto T1 nebo E1 rámců. Nejjednodušší způsob pochopení je představit si jeho použití při spojení dvou oddělených ústředen. Ty se často používají ve větších podnicích pro různá oddělení. Není žádoucí, aby si oddělení navzájem volaly přes externí telefonní linky. S tradičním zařízením, by bylo třeba mezi nimi natáhnout 25párový trunk nebo T1 linku. TDMoE emuluje E1, ale přenáší data v malé hlavičce uvnitř ethernetového rámce. To je umožňuje jen zařadit do stejného přepínače. Jakmile je TDMoE připojeno, Asterisk vidí jen 31 linek, které se zdají být E1 tudíž se nemusí vypořádávat se všemi složitostmi VoIP. TDMoE tedy sjednocuje flexibilitu a úspory nákladů Ethernetu s telefonními aspekty T1.

3.2 Tunelování přes Ethernet

Abychom mohli tunelovat E1 přes Ethernet rozhraní, je třeba nasměrovat rámce z trasy mezi Digium kartami do Ethernetového rozhraní. K tomu je potřeba opět upravit konfigurační soubor system.conf ve složce /etc/dahdi/. Do tohoto souboru vložíme pod nastavení karty řádek ve formě *dynamic=<driver>,<address>,<numchans>,<timing>,* kde *<driver>* je název užitého rozhraní na daném počítači (např. v našem případě eth). *<address>* je specifická adresa rozhraní (MAC adresa pro eth rozhraní), ovšem protější strany. To znamená, že v nastavení pro PC1 napíšeme MAC adresu Ethernetového rozhraní PC2 a naopak. *<numchans>* udává počet kanálů, při použití E1 nastavíme počet kanálů na 31.

Jako poslední nastavíme zdroj časování <*timing*> stejně jako jsme to provedli u nastavení telefonních karet Digium TE110P. Pokud zde nastavíme nulu, přikážeme tím systému, že dané PC se bude chovat jako Master. Pokud nastavíme jedničku, udáváme tím příkaz, že dané PC se bude chovat jako Slave. Jiné možnosti načasování zde možné nejsou.

Ve výsledku tedy bude vypadat celá konfigurace pro PC1 takto:

```
# Span 1: WCT1/0 "Digium Wildcard TE110P T1/E1 Card 0" (MASTER)
CCS/HDB3/CRC4

span=1,1,0,ccs,hdb3,crc4

dynamic=eth,eth3/1C:6F:65:C8:01:8E,31,1

# termtype: te

bchan=1-15,17-31

dchan=16

echocanceller=mg2,1-15,17-31
```

A pro PC2 bude vypadat konfigurace takto:

```
# Span 1: WCT1/0 "Digium Wildcard TE110P T1/E1 Card 0" (MASTER)
CCS/HDB3/CRC4

span=1,1,0,ccs,hdb3,crc4

dynamic=eth,eth2/1C:6F:65:3F:7A:84,31,0

# termtype: te

bchan=1-15,17-31

dchan=16

echocanceller=mg2,1-15,17-31
```

Rozhraní eth2 a eth3 jsou zde použita záměrně, protože na těchto rozhraních tyto počítače komunikují. Jak je vidět, nejdříve je třeba definovat obecné rozhraní, na kterém bude přenos probíhat a poté, za čárkou následuje konkrétní rozhraní daného počítače odděleného lomítkem od MAC adresy rozhraní protější strany. Více o této problematice viz**Chyba! Nenalezen zdroj odkazů.**[19]]

Po každé změně konfiguračního souboru system.conf je třeba jej uložit a restartovat DAHDI a Asterisk příkazy

```
#/etc/init.d/dahdi restart

#/etc/init.d/asterisk restart
```

Po jejich zadání proběhne odpojení a znovu načtení všech modulů uvedených v konfiguračním souboru modules.conf ve složce /etc/asterisk/.

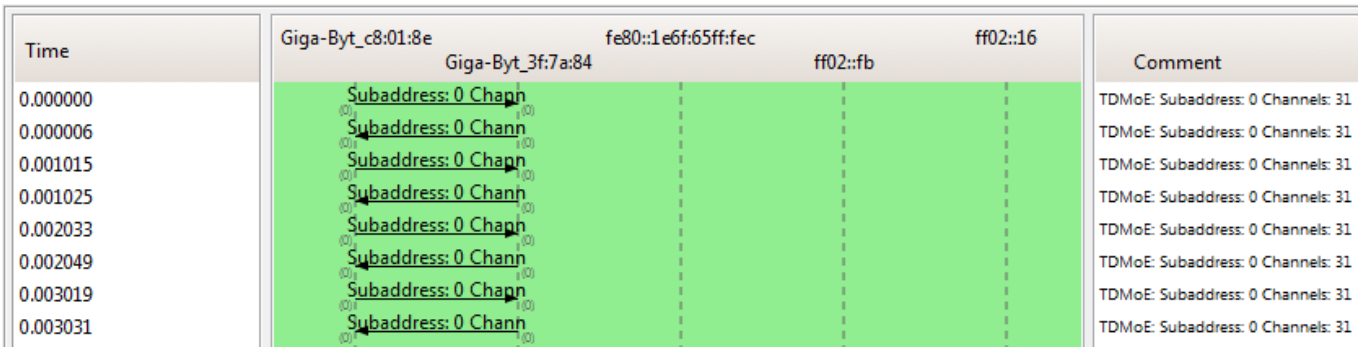
Pokud vše proběhlo správně, mněli bychom vidět v CLI Asterisku po zadání příkazu

```
*CLI>dahdi show status
```

další komunikační rozhraní pod již nakonfigurovanou kartou Digium TE110P, jako je vidět níže.

```
student-X58-USB3*CLI> dahdi show status
Description                              Alarms  IRQ  bpviol  CRC  Fra  Codi  Options
Digium Wildcard TE110P T1/E1 Card 0      OK      94   0       0    CCS  HDB3  CRC4
Dynamic 'eth' span at 'eth3/1C:6F:65:C8  UNCONFI 0    0       0    CAS  Unk
```

Na dalším obrázku 3.2 je vidět graf komunikace mezi oběma PC. Z tohoto grafu můžeme vyčíst směrování rámců, zdrojovou a cílovou adresu MAC zařízení, čas začátku přenosu rámce, užitý protokol a využité kanály.



Obrázek 3.2: *graf komunikace*

Protokol TDMoE byl původně vytvořen Markem Spencerem, zakladatelem Asterisku. Důvodem vývoje tohoto protokolu bylo umožnit uživatelům Asterisku používat TDM protokoly po levném hardwaru (Hardware Ethernet). Tento protokol je kontejner pro "všechny" TDM telefonní protokoly. Naneštěstí, protokol TDMoE neobsahuje informace o tom, co protokol (y) přenáší, takže abychom byli schopni dekódovat daný datový tok, musíme přesně vědět konfiguraci předávající strany. Uvnitř TDM proudu můžeme mít řadu různých konfigurací kódování a protokolů, jako jsou například ty, obsažené uvnitř BRI, PRI nebo SS7 signalizace.[20]

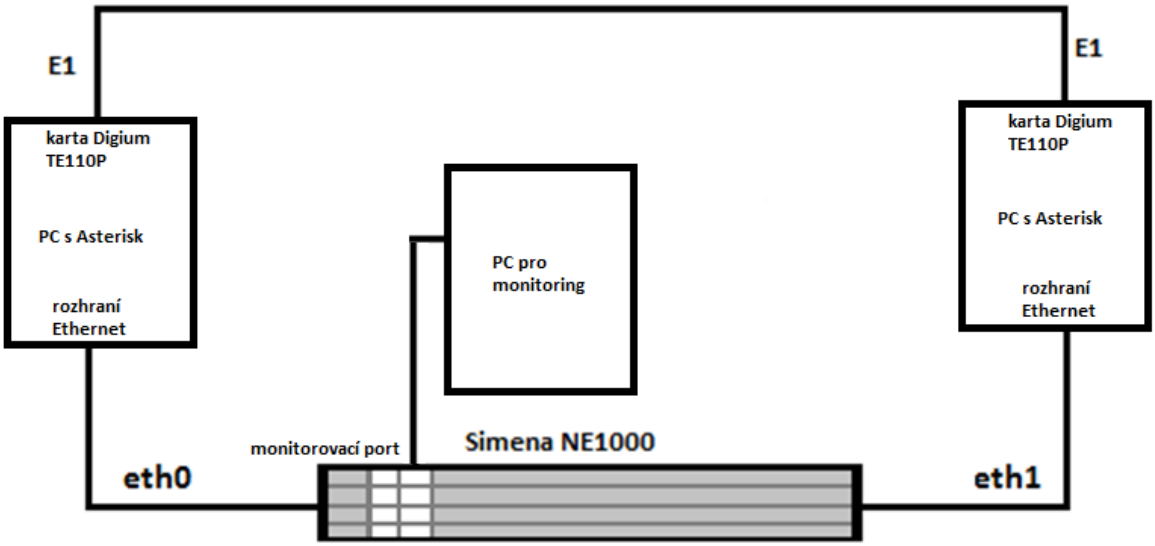
Po odchycení tohoto protokolu v programu Wireshark (pouze od verze 1.6 a vyšší protože tento protokol byl zaveden až v této verzi a do nižších verzí je třeba jej implementovat dodatečně) a vyexportování do textového souboru, si můžeme prohlédnout detaily každého rámce (vypsáno níže).

Jako první vidíme pořadí daného rámce (Frame 11), čas jeho příchodu (0.004998) v sekundách a zdrojovou MAC adresu a cílovou adresu, použitý protokol (TDMoE), jeho délku v bytech (286) a počet užitých kanálů (31) a jeho velikost (2288 bits).Na dalším řádku je vidět využité rozhraní (Ethernet II), pojmenování obou těchto použitých rozhraní na obou PC a jejich MAC adresy. Toto rozdělení se dá dále upřesnit na to, které rozhraní s kterou MAC adresou je zdrojem (Source) a které je cílem (Destination). Dále vidíme použitý protokol tedy Digium TDM over Ethernet Protocol . Ten lze také dále upřesnit a níže vidíme, zdali byl na trase indikován žlutý alarm (Yellow Alarm: False). Také zde vidíme počet užitých kanálů (Channels: 31). Jako poslední zde vidíme přenesená zašifrovaná data rozdělená po osmi bitech do jednotlivých Timeslotů.

```
No.    Time        Source                Destination
11     0.004998    (1c:6f:65:c8:01:8e)   (1c:6f:65:3f:7a:84)
Protocol      Length        Info
TDMoE         286           Subaddress:0 Channels:31
Frame 11: 286 bytes on wire (2288 bits), 286 bytes captured
(2288 bits)
Destination: Giga-Byt_3f:7a:84 (1c:6f:65:3f:7a:84)
Source: Giga-Byt_c8:01:8e (1c:6f:65:c8:01:8e)
Type: Digium TDM over Ethernet Protocol (0xd00d)
Digium TDMoE Protocol
Subaddress: 0
Samples: 8
Flags: 0x02, Sig bits present
....  ....0 = Yellow Alarm: False
....  ...1. = Sig bits present: True
Channels: 31
Data (8 bytes)
```

4 Měření kvalitativních parametrů vytvořeného spojení

Abychom změřili kvalitu vytvořeného spoje TDMoE, použijeme síťový emulátor Simena NE1000, do kterého zapojíme oba Ethernetové porty dle schématu na obrázku 4.1. Schéma zapojení. Jeden počítač tedy zapojíme do portu E1 a druhý do portu E1. Dále je potřeba připojit k síťovému emulátoru další počítač do monitorovacího portu označeného písmenem M a nastavit oba přístroje tak, aby byly ve stejné síti. Poté už stačí jen spustit webový prohlížeč a přihlásit se k webovému rozhraní Simeny. Propojení karet mezi sebou, jak je znázorněno na obrázku je možné z toho důvodu, že se jedná o synchronní přenos TDM a postačí nám sledovat stav tohoto rozhraní v Asterisku. V případě, že dojde k narušení synchronizace, bude tento stav detekován.



Obrázek 4.1: *Fyzické Schéma zapojení*

Měření probíhalo tak, že jsem nejprve hledal limitní podmínky pro provoz při jednotlivých zatíženích trasy BER, Jittrem a latencí. Poté jsem tyto jednotlivé parametry kombinoval mezi sebou za účelem zjištění jejich vlivu na kvalitu linky.

4.1 Alarmy

Kvalita trasy je odvozena od zobrazování alarmů příslušných pro karty Digium. Pokud je vše v pořádku a trasa je prostupná, uvidíme po zadání dahdi show status v *CLI> asterisku v sekci Alarms OK nebo UNCONFI jako na obrázku 4. 2. dahdi show status dole. UNCONFI v tomto případě naznačí chybu, ale pouze informuje o skutečnosti, že E1 není specificky nastavena a přenáší se celá ve svém základním stavu. To znamená, že nekontroluje pozici

synchronizačních kanálů. Mimo toto zobrazení, zde můžeme vidět indikované alarmy, jako jsou YEL, RED nebo BLUE (AIS).

4.1.1 YELLOW Alarm

Tento YEL nebo také RAI (Remote Alarm Indication - dálková signalizace alarmu) Alarm je na E1 portu generován tehdy, když obdrží signál od opačné strany trasy, kdy je tato opačná strana trasy v červeném alarmu. To v zásadě znamená, že opačná strana není schopna udržet synchronizaci, nebo nezískává námi vysílaný signál.

4.1.2 RED Alarm

Dalším alarmem, který lze vidět je RED Alarm. Ten se zobrazí, když se E1 port snaží udržet synchronizaci s opačným koncem trasy. Červený alarm označuje buď fyzický problém zapojení, ztrátu spojení nebo rámování, nebo pokud linkové kódování nesouhlasí s opačnou stranou. Zároveň vysílá žlutý alarm po trase do cíle na opačné straně pro indikaci problému s příjímáním signálu. Alarm lze vidět ve zdrojovém kódu níže.

4.1.3 BLUE Alarm

E1 port bude indikovat modrý alarm, když příjmá všechny timesloty zpožděné ze vzdálené strany. Tento speciální signál naznačuje, že vzdálená strana má problémy vysíláním dat. Alarm signalizuje problém proti proudu dat a to od switche, ke kterému jsme připojeni.[21]

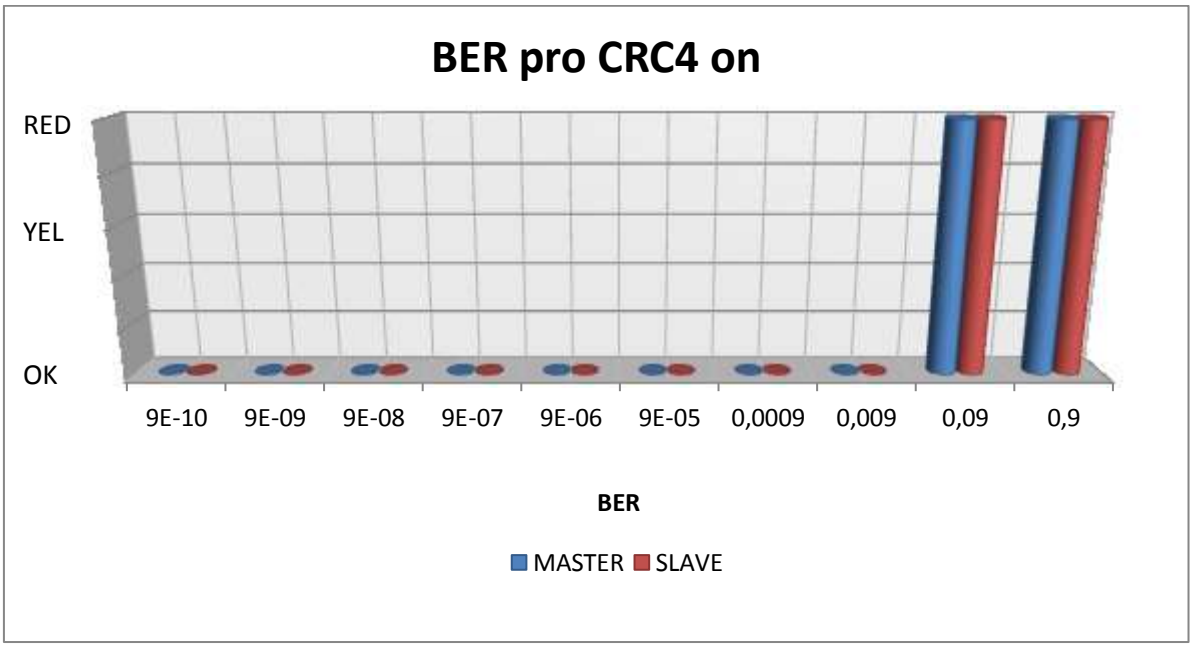
```
student-X58-USB3*CLI>  dahdi show status
Description           Alarms  IRQ  bpviol CRC Fra  Codi  Options
Digium Wildcard TE110P OK     94   0      0   CCS  HDB3     CRC4
Dynamic eth3/1C:6F    UNCONFI  0   0      0   CAS  Unk
student-X58-USB3*CLI>  dahdi show status
Description           Alarms  IRQ  bpviol CRC Fra  Codi  Options
Digium Wildcard TE110P RED    94   0      0   CCS  HDB3     CRC4
Dynamic eth3/1C:6F    UNCONFI  0   0      0   CAS  Unk
```

Nejprve jsem provedl měření pro zapnutý opravný kód CRC4. Toto nastavení si lze ověřit v příkazovém řádku asterisku *CLI> po zadání příkazu dahdi show status, v sekci Options jak je vidět na kódu odchyceném přes putty výše.

4.1 Měření Bit Error Rating

Tabulka 4.1: *Bit Error Rating*

BER	*ALARM MASTER*	*ALARM SLAVE*
$1 \cdot 10^{-9}$	OK	OK
$1 \cdot 10^{-8}$	OK	OK
$1 \cdot 10^{-7}$	OK	OK
$1 \cdot 10^{-6}$	OK	OK
$1 \cdot 10^{-5}$	OK	OK
$1 \cdot 10^{-4}$	OK	OK
$1 \cdot 10^{-3}$	OK	OK
$1 \cdot 10^{-2}$	RED	RED
$1 \cdot 10^{-1}$	RED	RED



Obrázek 4.2: *BER pro CRC4 on*

Na obrázku 4.2 BER pro CRC4 on vidíme závislost kvality trasy na hodnotách BER jak pro master stranu, tak pro Slave stranu. Z grafu je vidět, že trasa je schopna bezproblémového provozu, dokud BER, nedosáhne hodnoty 1·10-2, kdy se trasa stává nepoužitelnou.

4.2 Měření Latence

V příloze A, *Latence CRC4 on* vidíme závislost kvality trasy na délce latence jak pro master stranu tak pro Slave stranu. Z grafu je vidět, že trasa je schopna bezproblémového provozu dokud Latence nedosáhne hodnoty 1200 milisekund na Master straně, kdy se poprvé objeví RED alarm a trasa se tak stává nepoužitelnou. Vzhledem k obrovské stabilitě trasy jsem přesunul tuto část měření do příloh.

4.3 Měření Jitteru

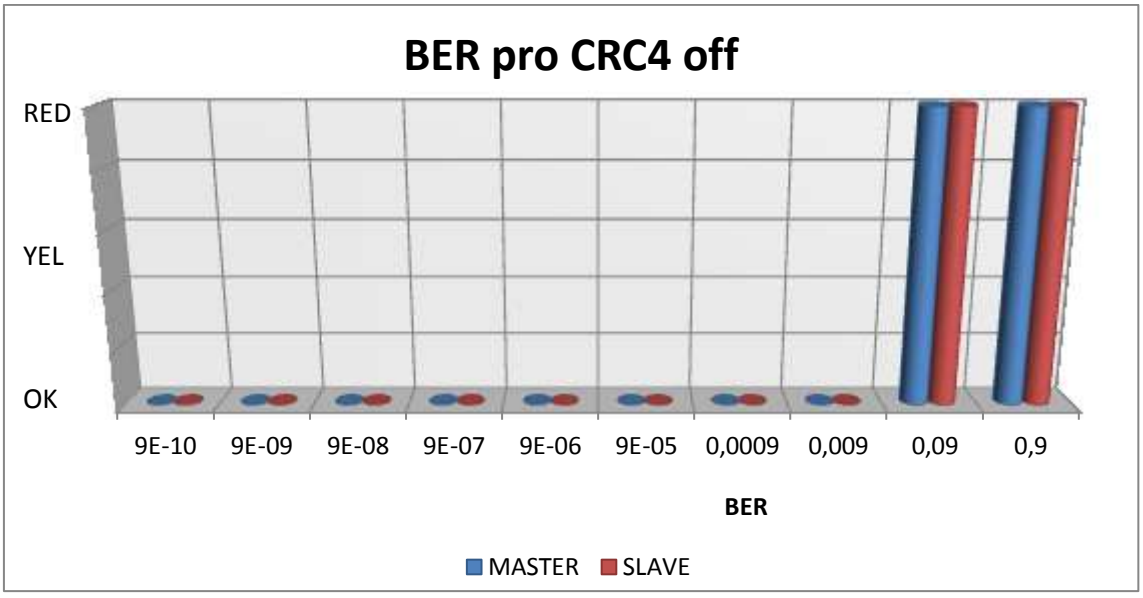
V příloze B, *Jitter CRC4 on* vidíme závislost kvality trasy na velikosti Jitteru, jak pro master stranu tak pro Slave stranu. Z grafu je vidět, že dokud nedosáhneme jitteru 1500 milisekund, trasa je schopna provozu. Od této hodnoty je na Master straně generován alarm RED a na straně Slave je od této hodnoty generován alarm YEL.

Stejná měření jsem provedl i při vypnutém opravném kódu CRC4, se stejným postupem.

4.4 Měření Bit Error Rating pro CRC4 off

Tabulka 4.2: *Naměřené hodnoty BER*

BER	ALARM MASTER	ALARM SLAVE
$1 \cdot 10^{-9}$	OK	OK
$1 \cdot 10^{-8}$	OK	OK
$1 \cdot 10^{-7}$	OK	OK
$1 \cdot 10^{-6}$	OK	OK
$1 \cdot 10^{-5}$	OK	OK
$1 \cdot 10^{-4}$	OK	OK
$1 \cdot 10^{-3}$	OK	OK
$1 \cdot 10^{-2}$	RED	YEL
$1 \cdot 10^{-1}$	RED	YEL



Obrázek 4.3: *BER pro CRC4 off*

Na obrázku 4.3 BER pro CRC4 off vidíme závislost kvality trasy na hodnotách BER jak pro master stranu tak pro Slave stranu. Z grafu je vidět, že trasa je schopna bezproblémového provozu dokud BER nedosáhne hodnoty 1·10-2, kdy se trasa stává nepoužitelnou.

Po porovnání s měřením se zapnutým kódem CRC4 můžeme usoudit, že tento kód nemá vliv na kvalitu trasy, pokud je tato trasa zatěžována pouze chybou Bit Error Rating.

4.5 Měření latence

Vzhledem k tomu, že hodnoty jsou nad úrovní kvality přenosu řeči a tudíž nejsou pro měření kvality trasy pro přenos E1 důležité, jsou přesunuty do přílohy C, *Latence pro CRC4 off*. V příloze je vidět vliv latence na kvalitu vytvořené trasy. Trasa je schopna fungovat bezproblémově do latence 1400 milisekund, kdy strana Slave začne generovat YEL alarm a strana Master začne ve stejnou chvíli generovat alarm RED. K absolutnímu krachu trasy dochází kolem 2000 ms, kdy začne RED alarm generovat i strana Slave.

4.6 Měření Jitteru

V příloze D, *Měření Jitteru pro CRC4 off* vidíme, jakým způsobem je zkonstruovaná trasa ovlivněna Jittrem. Dokud se Jitter pohybuje do hodnoty 1000 milisekund, je trasa schopná bezproblémového provozu. Nad hranici 1100 milisekund se trasa postupně deformuje a jsou vidět generované alarmy. Nejdříve generuje Master strana žlutý alarm značící problém

s timingem a krátce poté červený alarm značící rozpad spojení. Ovšem strana Slave generuje problem s timingem až kolem 1500 milisekund, nicméně pro bezproblémový chod trasy je limitních 1000 milisekund, kdy Master strana přestává vidět protistranu.

4.7 Porovnání Jitteru při zapnutém a vypnutém CRC4

V příloze E, *Porovnání Jitteru při zapnutém a vypnutém opravném kódu* je vidět vliv opravného kódu CRC na kvalitu trasy. Při jeho vypnutí začala Master strana generovat rozpad spojení o 200 milisekund dříve než když byl CRC4 kód zapnutý.

Zpoždění v komunikaci mezi Master a Slave se projevovalo přesně podle hodnoty nastavené na Jitteru, podle této hodnoty trvalo přenesení alarmu mezi Master stranou a jeho objevením se na straně Slave. V praxi se tento jev projevoval tak, že při nastavení Jitteru na 1000 ms se objevil alarm nejdříve na straně Master a o sekundu později na straně Slave.

Dalším krokem byla kombinace těchto faktorů a jejich vliv na kvalitu trasy, když působí společně.

4.8 Vliv Jitteru a latence na kvalitu trasy

V příloze F, *Vliv Jitteru a Latence na kvalitu trasy při zapnutém CRC4* je vidět, co se stane s trasou, pokud zvyšujeme latenci po sto milisekundách a navíc trasu zatěžujeme Jittrem. Nejdéle je trasa schopná provozu při co nejmenší latenci. Při její stoupající hodnotě se kvalita trasy zhoršuje téměř lineárně. Vzhledem k přívětivým výsledkům při zatěžování trasy pouze Jittrem, má latence drastický vliv na zhoršování schopnosti trasy přenášet data.

Totožné měření jsme provedli i s vypnutým opravným kódem, které je vidět na obrázku 4.11. Pokud tyto dvě měření porovnáme, je patrný markantní vliv opravného kódu na provozuschopnost trasy. Při jeho vypnutí se trasa zhoršila téměř trojnásobně. Proto je ve všech doporučeních pro přenos TDM přes Ethernet uvedeno užití tohoto kódu.

4.9 Měření latence v jednom směru trasy

Pro bližší specifikaci vlivu latence na kvalitu trasy jsem tuto chybu proměřil i v jednom směru. Nejprve ve směru Master -> Slave a poté v opačném Slave -> Master. Z níže uvedených grafů je patrné, jak důležitý je zdroj časování pro kvalitu trasy. Pokud nejdříve vysílal Master, dosahovala trasa výborných výsledků a krach spojení se projevil až při 18000 milisekundách.

Ovšem jakmile jsem směr otočil a nejdříve vysílal Slave, který je závislý na generování hodinového signálu Master strany, byla trasa nepoužitelná již při 1700 milisekundách, jak je

patrné v příloze G. Stejné měření jsem provedl i při vypnutém opravném kódu, kdy byl výsledek měření totožný. Tudíž lze usoudit, že opravný kód nemá vliv na kvalitu trasy při jejím zatěžování latencí v jednom směru.

Ojediněle se vyskytly alarmy i u nižších hodnot jako 2500 milisekund nebo 6100 milisekund, ale po čtyřech opakovaných měřeních se tyto výskyty nepodařilo zopakovat. Přisuzuji je tedy náhodné chybě měření a neuvažuji s nimi ve výsledných grafech. Opakovaně nastal žlutý alarm při hodnotě 15000 milisekund Na straně master i Slave a červený alarm značící krach trasy nastal při 18000 milisekund.

Při měření v opačném směru je vidět markantní zhoršení schopnosti trasy přenášet data oproti opačnému směru Master -> Slave. I tak jsou ovšem hodnoty více než optimistické. To proto, že se nejedná o synchronní protokol s potvrzováním. Proto můžou být hodnoty zpoždění tak velké.

Jak je ze závěrů patrné, tyto hodnoty dalece přesahují hodnoty udržitelnosti E1. Důvodem k takto extrémním hodnotám je podstata odděleného vysílání a příjmu signálu.

4.10 Bandwidth

Po odchycení přenosu z wiresharku můžeme vidět, kolik rámců je přeneseno za sekundu. Jelikož se tento počet mění, pro představu průměrného toku rámců si odvodíme, kolik rámců se přenese za jednu minutu a poté vydělíme 60 viz. příklad 4.1. Z wiresharku jsme zjistili, že za minutu se přenese 120 168 rámců, tudíž

$$\frac{120168}{60} = 2002{,}8 \text{ rámců za sekundu} \tag{4.1}$$

Z tohoto výpočtu lze dále odvodit, jak často se v průměru posílá jeden rámec. Počet rámců převezmeme z předchozího příkladu a budeme uvažovat čas tisíc milisekund.

$$\frac{1000}{2002{,}8} = 0{,}499 \text{ ms} \approx 499\mu\text{s} \tag{4.2}$$

pakety se tedy vysílají co 499 mikrosekund.

Z výpočtu 4.1 lze dále odvodit přenosovou rychlost rámce. Protože je velikost rámce 2288 bitů a my víme, že za jednu sekundu je přeneseno v průměru 2002,8 rámců, výpočtem 4.3 zjistíme jeho přenosovou rychlost.

$$2002{,}8 \cdot 2288 = 4582406{,}4 \text{ bitů za sekundu} \approx 4{,}58 \text{ Mbit/s} \tag{4.3}$$

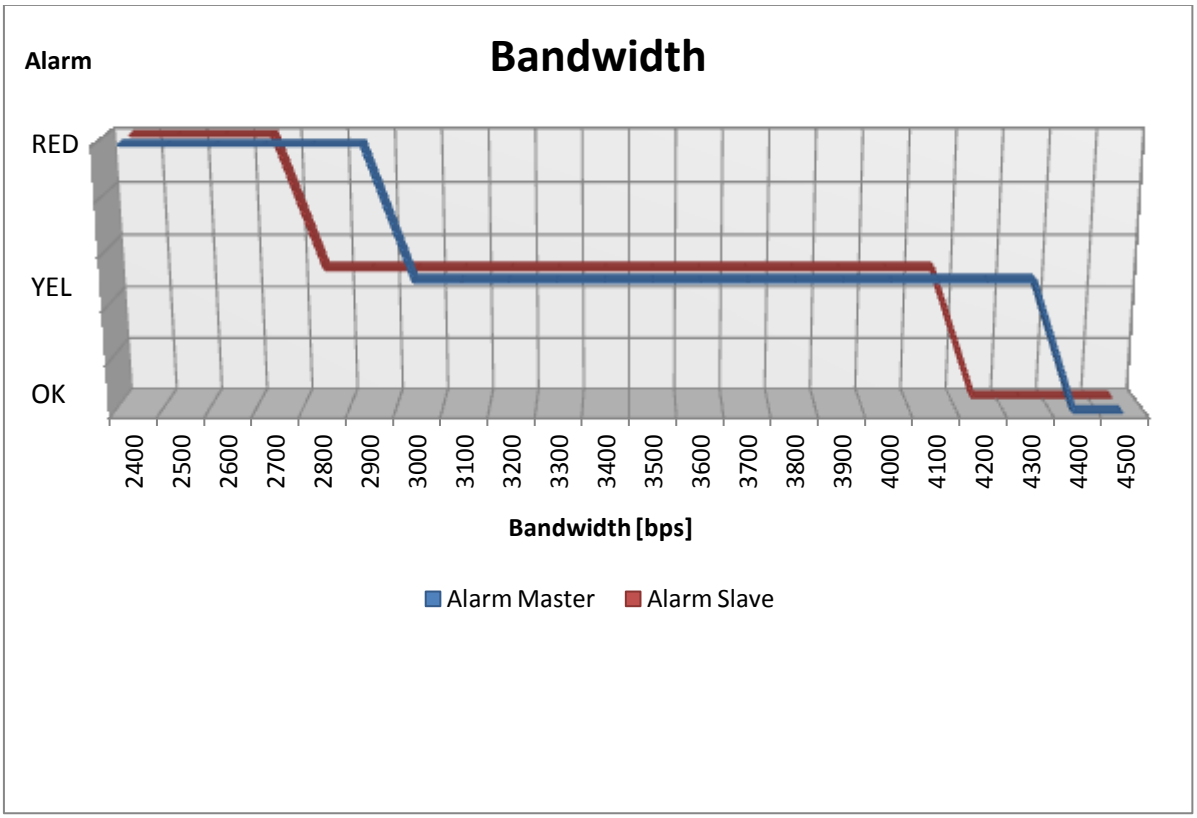
Z tohoto výpočtu vyplívá, že síťový emulátor Simena pracuje v plném duplexu a rámce jsou přenášeny průměrnou rychlostí 4,58Mbit/s. Tato rychlost se pak při zatěžování trasy jednotlivými negativními vlivy snižovala až na polovinu.

Dále si také můžeme odvodit velikost hlavičky rámce. Protože rámec má délku 2288 bitů a v každém tomto rámci je přenášeno 30 kanálových intervalů o velikosti 8 bajtů, použijeme následujícího výpočtu.

$$30 \cdot 8 \cdot 8 = 1920 \text{ bitů pro data} \tag{4.4}$$

$$2288 - 1920 = 368 \text{ bitů} \tag{4.5}$$

Pro data je tedy v rámci rezervováno 1920 bitů a hlavička rámce tedy zabírá 368 bitů.



Obrázek 4.4: *Bandwidth*

Při měření s fixně se měnící šířkou pásma je vidět, že limitní šířka pásma pro přenos signálu po trase je 4300 bps, což je téměř dvojnásobek šířky pásma, které k přenosu potřebuje jeden TDMoE rámec.

Závěr

Cílem mé bakalářské práce bylo zkonstruovat trasu schopnou přenosu časového multiplexu přes Ethernet. K tomu jsem využil open-source Asterisk 1.8 v kombinaci s telefonními kartami Digium TE110P. Takovýto způsob řešení jsem použil kvůli možnosti využít specifického protokolu TDMoE vytvořeného Markem Spencerem, původním vývojářem Asterisku a také kvůli jeho minimálním pořizovacím nákladům, kdy je potřeba koupit pouze telefonní karty. Toto řešení je výhodné v tom, že přímo nad Ethernetovým rámcem je přenášen tento TDMoE protokol, bez využití TCP nebo UDP protokolu, jak je vidět na výpisu z Wiresharku. V praktické části je popsána realizace a způsob nastavení tohoto řešení. V další části jsem zatěžoval trasu primárními kvalitativními parametry jako je Latence, Jitter, BER nebo změna šířky pásma Bandwidth. Tyto jednotlivé měření jsem dále zpracoval do tabulek a přehledných grafů, kde je vidět do kdy je trasa schopná provozu, než začne indikovat jednotlivé alarmy značící problémy na trase nebo její rozpad.

Na grafech je vidět, že pro funkcionalitu trasy je limitní hodnota BER $1 \cdot 10^{-3}$, protože od hodnoty $1 \cdot 10^{-2}$ je již generován RED alarm na obou koncích trasy. Stejně tak je při zatížení trasy Jittrem limitní hodnota 1500 milisekund, kdy jsou opět na obou stranách trasy generovány alarmy indikující její rozpad. Jakmile trasu zatížíme Latencí, dostaneme se na limitní hodnotu pro přenos 1200 milisekund, od které jsou opět generovány alarmy. Pokud povolíme pouze směr od zdroje časování k druhému konci (směr Master -> Slave) dostáváme se až k limitním hodnotám 18 000 milisekund. Při opačném směru (tedy Slave -> Master) ovšem trasa krachuje již při 1600 milisekundách, z čehož se dá odvodit, že hlavní vliv na robustnost trasy má přenos paketů ve zpětném směru. Při testování potřebné šířky pásma bandwidth jsem se dostal k limitním podmínkám při 4300 bps. Což je téměř dvojnásobná hodnota, než kterou potřebuje jeden TDMoE rámec.

Tato měření jsem provedl i pro vypnutý opravný kód CRC4. Kód neměl vliv na kvalitu trasy při jejím zatížení BER, Latencí v jednom směru nebo bandwidth, kdy byly naměřené výsledky téměř totožné s výsledky při zapnutém CRC4. Zatímco u Jitteru můžeme pozorovat pokles kvality trasy o necelých 27%.

Z těchto naměřených parametrů je vidět robustnost trasy, kdy její kvalitativní parametry několikanásobně převyšují limitní podmínky pro přenos hlasu. Kupříkladu maximální možné zpoždění pro přenos hlasu, aby byl bez problémů srozumitelný pro druhou stranu, je do 300 milisekund. Ovšem hodnota, při které se rozpadá trasa je 1200 milisekund. Proto bych neváhal nasadit přenos TDM přes ethernet do reálného provozu.

Použitá literatura

[1] Netem. Linux Foundation [online]. 2014 [cit. 2014-05-03]. Dostupné z: http://www.linuxfoundation.org/collaborate/workgroups/networking/netem

[2] ŠKORPIL, CSC., Ing. Vladislav. Rozhraní E1. Elektrorevue: Základy telekomunikační hierarchie [online]. 1999 [cit. 2014-04-30]. Dostupné z: http://www.elektrorevue.cz/clanky/99015/index.html

[3] ITU-T G.703. Physical/electrical characteristics of hierarchical digital interfaces. Geneva: ITU, 2001.

[4] ITU-T G.704. Synchronous frame structures used at 1544, 6312, 2048, 8448 and 44 736 kbit/s hierarchical levels. ITU, 1998.

[5] Kód HDB3 (HDBn). Encyklopedie.amapro [online]. [cit. 2014-05-03]. Dostupné z:http://amapro.cz/encyklopedie/digitalni_technika/kod%20hdb3%20(hdbn).php

[6] Q.933. ISDN Digital Subscriber Signalling System No. 1 (DSS1) – Signalling specifications for frame mode switched and permanent virtual connection control and status monitoring. Geneva: ITU, 2003.

[7] Time-division Multiplexing. In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2014-05-04]. Dostupné z: http://en.wikipedia.org/wiki/Time-division_multiplexing

[8] Simena. Hywire associates [online]. [cit. 2014-05-04]. Dostupné z: http://hywireassociates.com/img/131110.pdf

[9] Speech and multimedia Transmission Quality (STQ); Quality Assessment of Synthesized Speech. Http://www.etsi.org/deliver/etsi_tr/102900_102999/102948/01.02.01_60/tr_102948v010201p.pdf [online]. 2012 [cit. 2014-04-30]. Dostupné z: http://www.etsi.org/deliver/etsi_tr/102900_102999/102948/01.02.01_60/tr_102948v010201p.pdf

[10] HAßLINGER, Gerhard a Oliver HOHLFELD. The Gilbert-Elliott Model for Packet Loss in Real Time Services on the Internet. [online]. 2008 [cit. 2014-05-04]. Dostupné z: http://www.ohohlfeld.com/paper/hasslinger_hohlfeld-mmb_2008.pdf

[11] ITU-T G.823. The control of jitter and wander within digital networks which are based on the 2048 kbit/s hierarchy. Geneva: ITU, 2000.

[12] VOZŇÁK, PH.D., Doc. Ing. Miroslav. TELEFONNÍ ÚSTŘEDNY ASTERISK. Odborný seminář [online]. 2008 [cit. 2014-04-30]. Dostupné z: http://www.ip-telefon.cz/archiv/dok_osta/ipt-2008_Telefonni_ustredny_Asterisk.pdf

[13] T1/E1/J1 Cards. OpenVox [online]. 2009 [cit. 2014-05-04]. Dostupné z: http://www.openvox.cn/en/products.html?category_id=3&page=shop.browse&limit=20&start=20

[14] Sangoma [online]. 2013 [cit. 2014-05-04]. Dostupné z: http://www.sangoma.com/

[15] Digium [online]. 2014 [cit. 2014-05-04]. Dostupné z: http://www.digium.com/en/

[16] DAHDI. Voip-info.org [online]. 2014 [cit. 2014-05-04]. Dostupné z: http://www.voip-info.org/wiki/view/DAHDI

[17] MADSEN, Lief, Jim VAN MEGGELEN a Russell BRYANT. Asterisk™: The Definitive Guide [online]. 3. vyd. 2011 [cit. 2014-04-30]. Dostupné z: http://www.asteriskdocs.org/en/3rd_Edition/asterisk-book-html-chunk/index.html

[18] NESSJØEN, Håkon. Asterisk, DAHDI and TDMoE [online]. 2010 [cit. 2014-05-04]. Dostupné z:http://lunatic.no/2010/02/asterisk-dahdi-and-tdmoe/

[19] VOZNAK M., BENES J., TDM over IP solution. In Proc. Research in Telecommunication Technologies (RTT2006), Brno University of Technology, 2006, p.382-384.

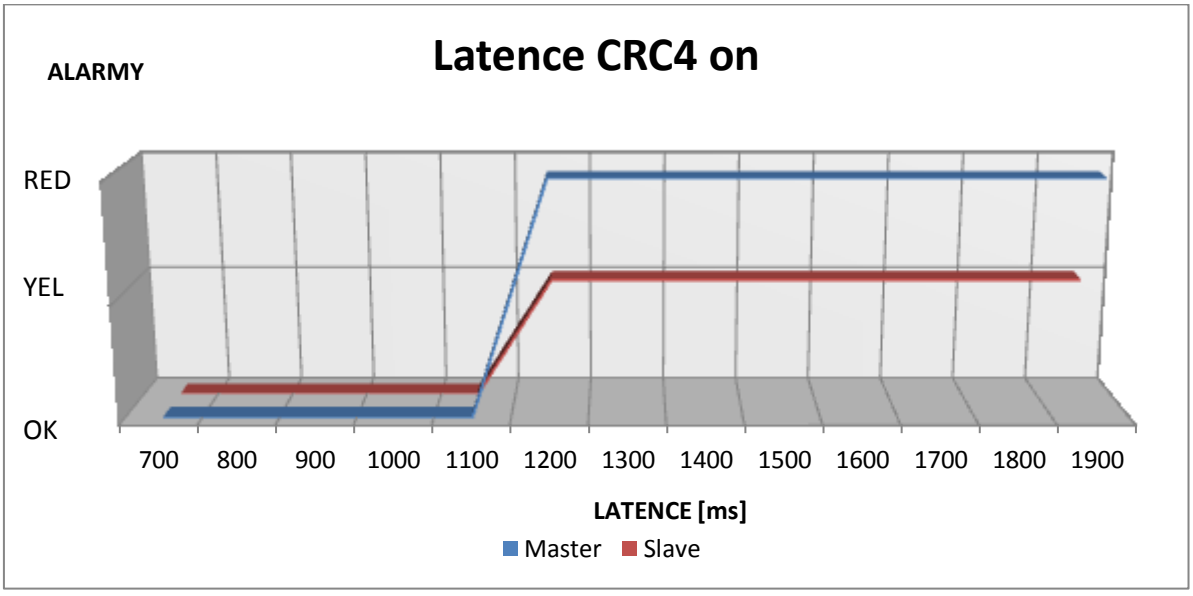
[20] TDMoE. Wireshark Wiki [online]. 2012 [cit. 2014-05-04]. Dostupné z: http://wiki.wireshark.org/TDMoE

[21] DAHDI Telephony Interface Driver. [online]. 2014 [cit. 2014-05-04]. Dostupné z: http://docs.tzafrir.org.il/dahdi-linux/README.html#_alarm_types

5 Seznam příloh

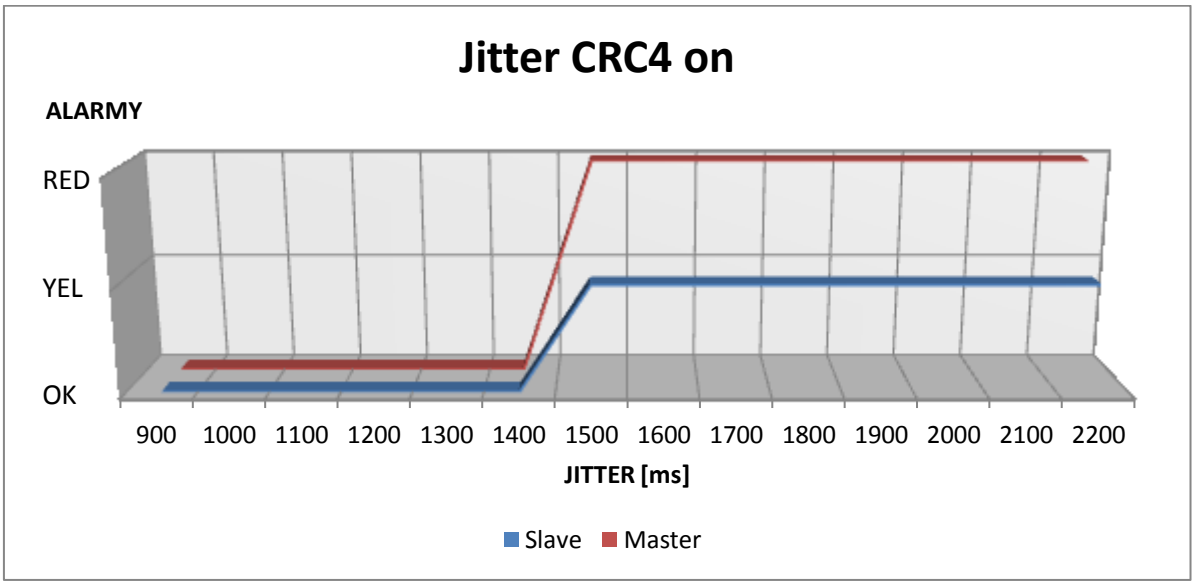
Příloha A: *Latence CRC4 on*

LATENCE fixní [ms]	ALARM MASTER	ALARM SLAVE
700	OK	OK
800	OK	OK
900	OK	OK
1000	OK	OK
1100	OK	OK
1200	RED	YEL
1300	RED	YEL
1400	RED	YEL
1500	RED	YEL
1600	RED	YEL
1700	RED	YEL
1800	RED	YEL
1900	RED	YEL



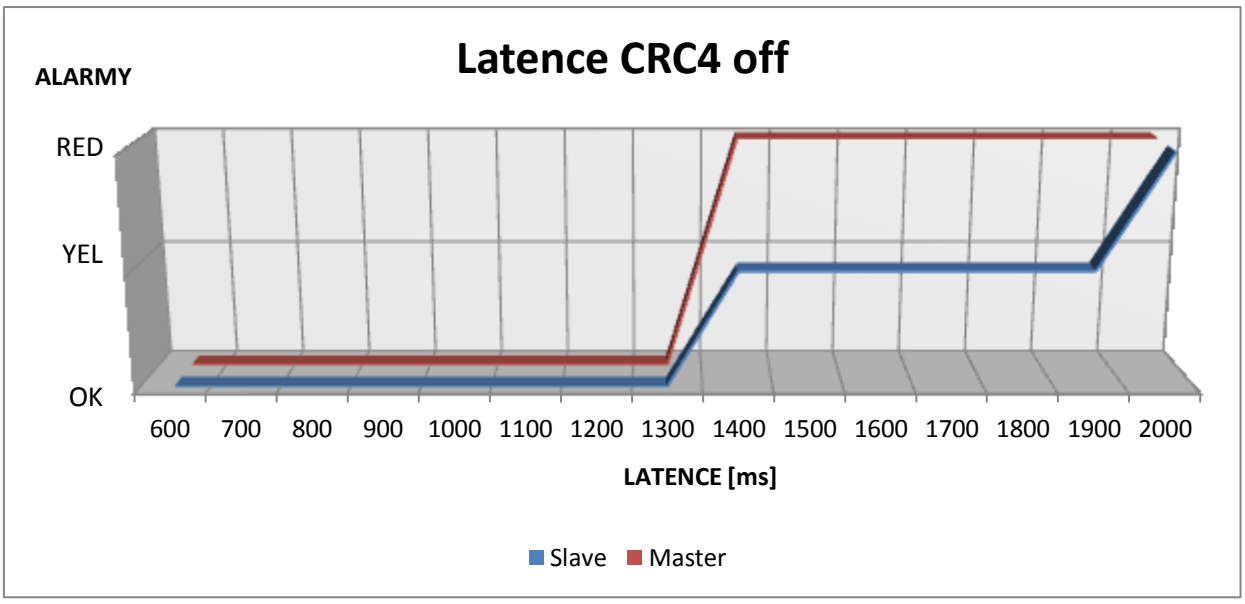
Příloha B: *Jitter CRC4 on*

JITTER [ms]	ALARM MASTER	ALARM SLAVE
900	OK	OK
1000	OK	OK
1100	OK	OK
1200	OK	OK
1300	OK	OK
1400	OK	OK
1500	RED	YEL
1600	RED	YEL
1700	RED	YEL
1800	RED	YEL
1900	RED	YEL
2000	RED	YEL
2100	RED	YEL
2200	RED	YEL



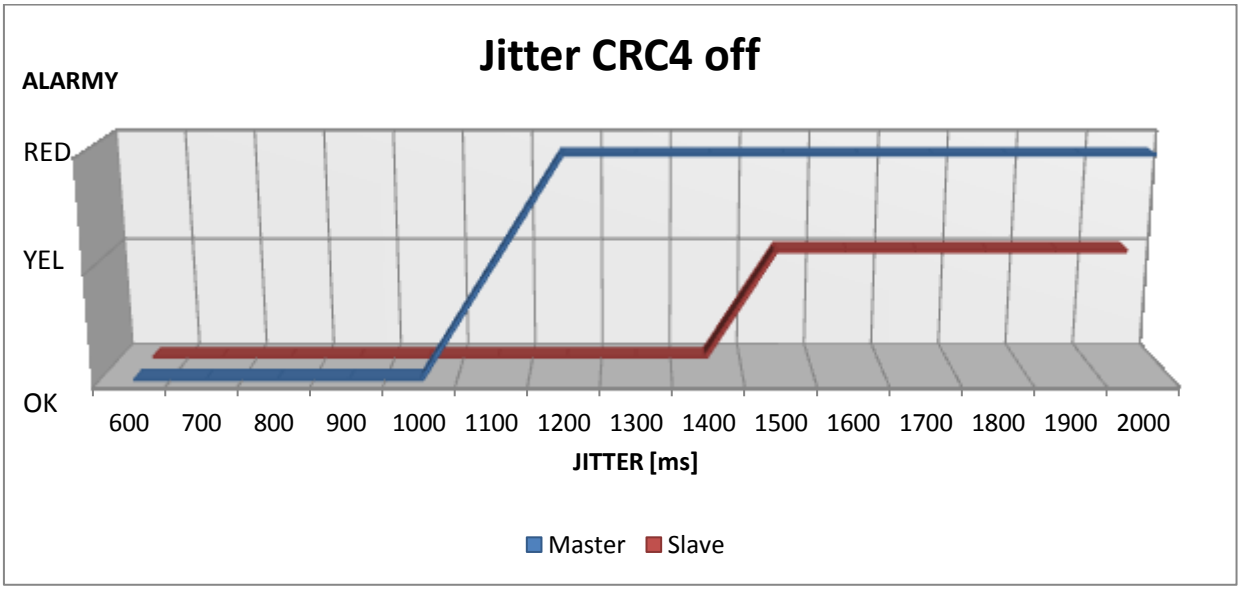
Příloha C: *Latence pro CRC4 off*

LATENCE fixní [ms]	ALARM MASTER	ALARM SLAVE
600	OK	OK
700	OK	OK
800	OK	OK
900	OK	OK
1000	OK	OK
1100	OK	OK
1200	OK	OK
1300	OK	OK
1400	RED	YEL
1500	RED	YEL
1600	RED	YEL
1700	RED	YEL
1800	RED	YEL
1900	RED	YEL
2000	RED	RED

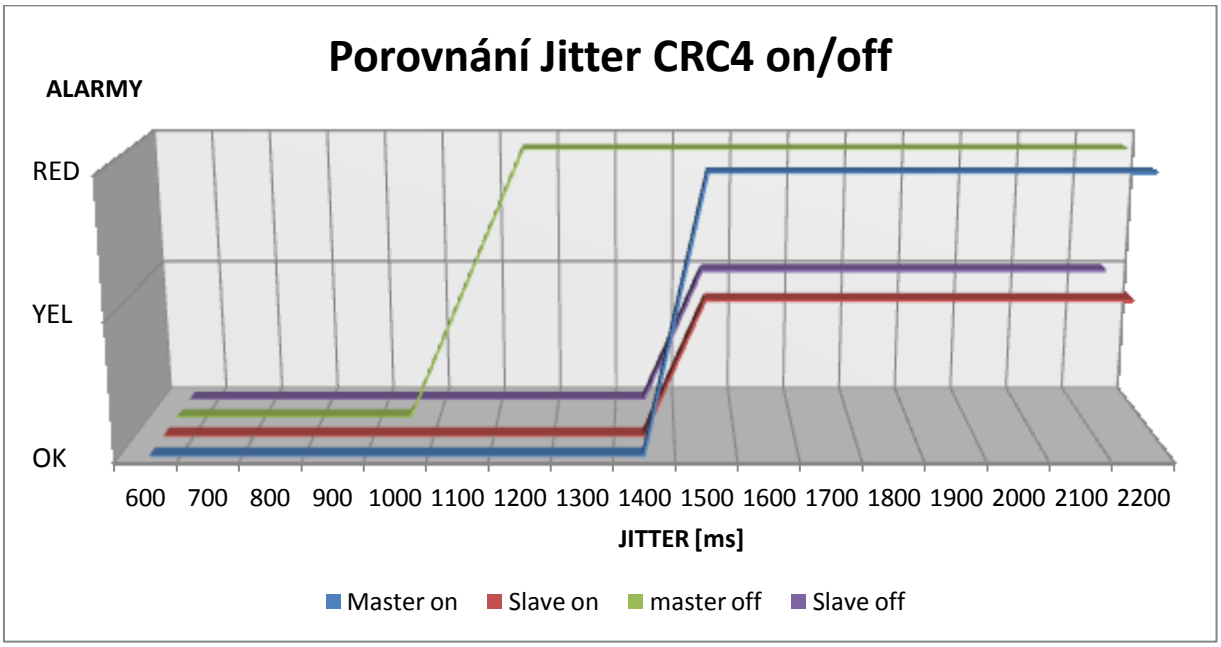


Příloha D: *Měření Jitteru pro CRC4 off*

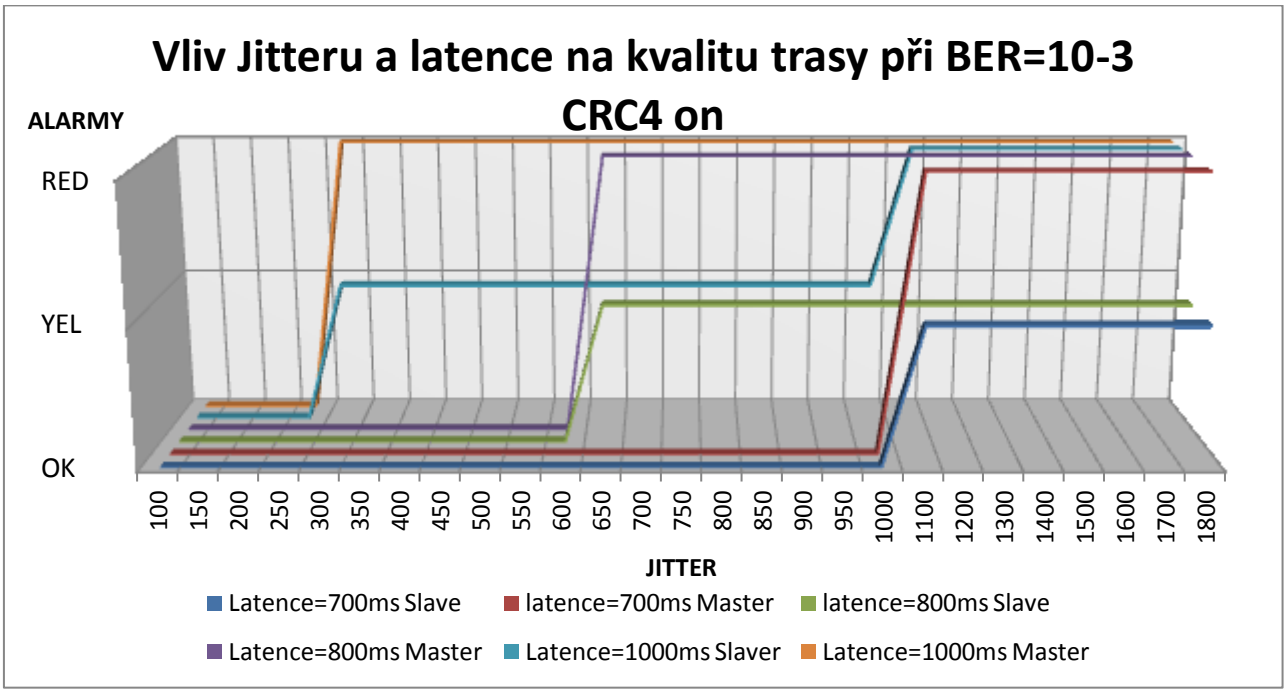
JITTER [ms]	ALARM MASTER	ALARM SLAVE
600	OK	OK
700	OK	OK
800	OK	OK
900	OK	OK
1000	OK	OK
1100	YEL	OK
1200	RED	OK
1300	RED	OK
1400	RED	OK
1500	RED	YEL
1600	RED	YEL
1700	RED	YEL
1800	RED	YEL
1900	RED	YEL
2000	RED	YEL

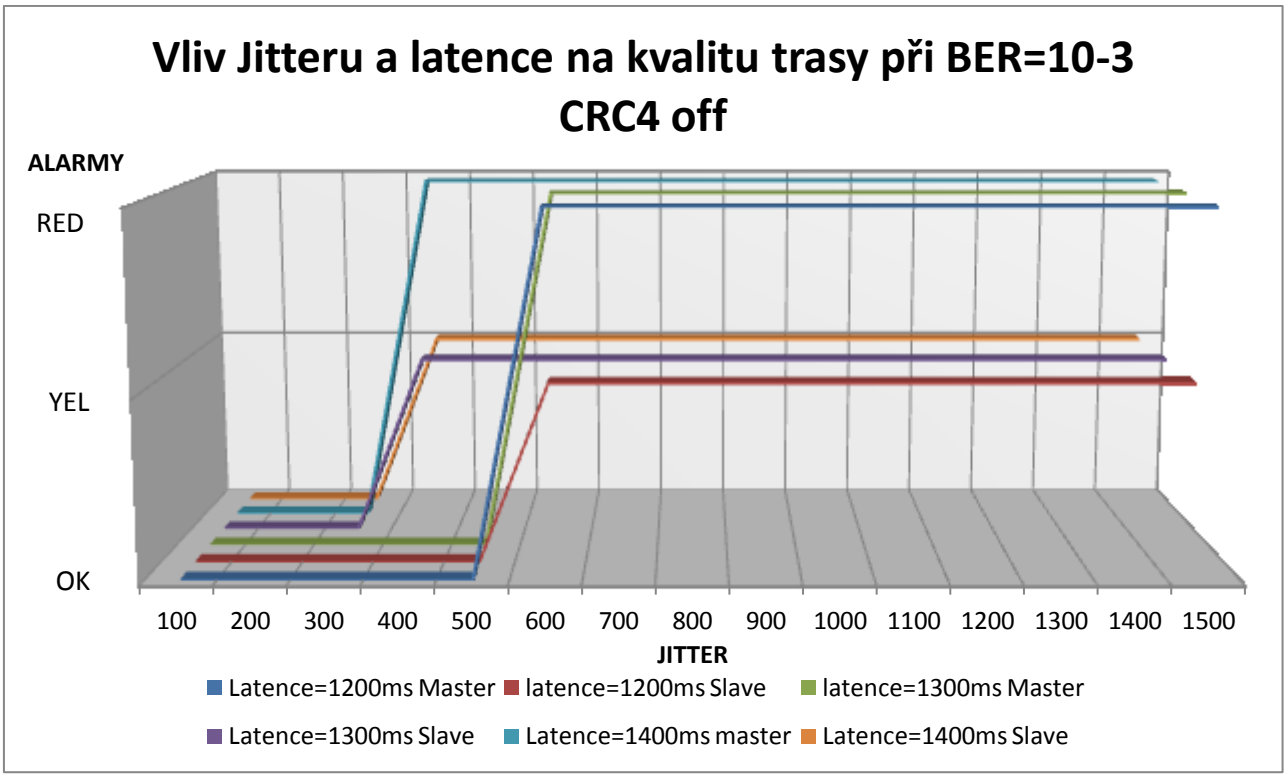


Příloha E: *Porovnání Jitteru při zapnutém a vypnutém opravném kódu*



Příloha F: *Vliv Jitteru a Latence na kvalitu trasy při zapnutém CRC4*

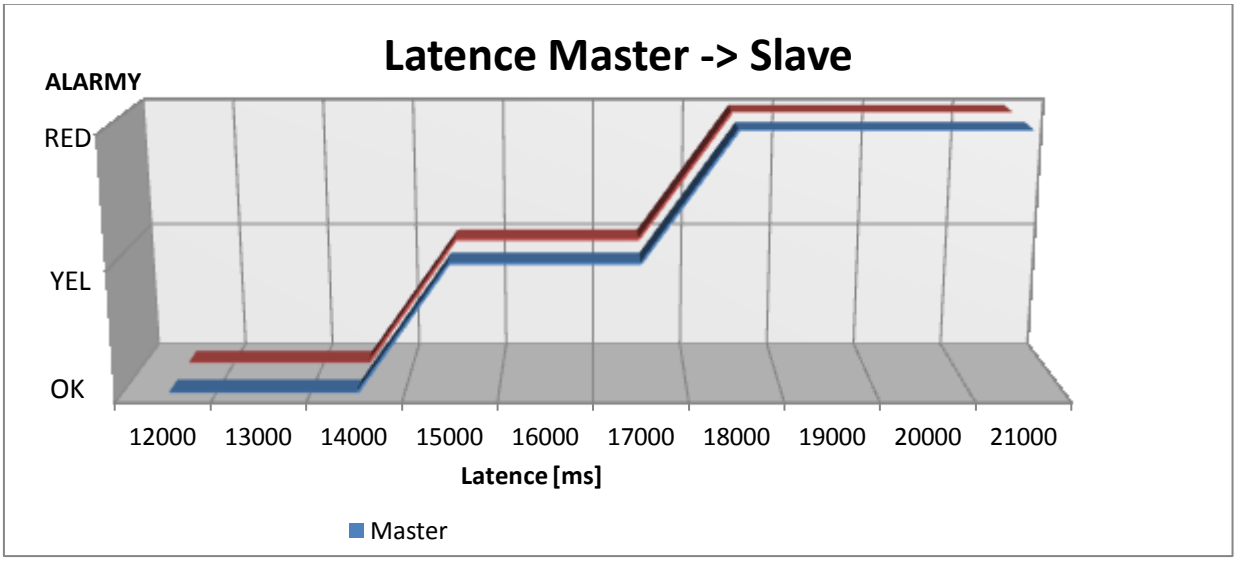




Příloha G: *Měření latence v jednom směru*

Latence ve směru Master ->Slave

Latence [ms] Master -> Slave	Alarm Master	Alarm Slave
12000	OK	OK
13000	OK	OK
14000	OK	OK
15000	YEL	YEL
16000	YEL	YEL
17000	YEL	YEL
18000	RED	RED
19000	RED	RED
20000	RED	RED
21000	RED	RED



/

Latence ve směru Slave ->Master

Latence [ms]	Alarm Master	Alarm Slave
1200	OK	OK
1300	OK	OK
1400	OK	OK
1500	OK	OK
1600	OK	YEL
1700	RED	YEL
1800	RED	YEL
1900	RED	YEL
2000	RED	YEL
2100	RED	YEL